Queen Mary University of London
School of Electronic Engineering & Computer Science

A CREDIT-BASED APPROACH TO SCALABLE VIDEO TRANSMISSION OVER A PEER-TO-PEER SOCIAL NETWORK

Thesis submitted to University of London
in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy

Stefano Asioli

Supervisor: Prof. Ebroul Izquierdo

London, 2013.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor Prof. Ebroul Izquierdo, as this work would not have been possible without his support and advice.

A special thank goes to Dr. Naeem Ramzan, for his help and above all for his patience.

During these last years I have had the chance to work with many brilliant people, too many to mention, nevertheless each one of them has left me something. I would like to thank, in no particular order: Eduardo Peixoto, Michele Sanna, Qianni Zhang, Lasantha Seneviratne, Giuseppe Passino, Muhammad Akram, Tijana Janjusevic, Bruno Gardlo, Sander Koelstra, Julie Wall, Krishna Chandramouli, Vlado Kitanovski, Heng Yang 'Chris', Ivan Damnjanovic, Saverio Blasi, Fiona Rivera, Tomas Piatrik, Virginia Fernandez, César Pantoja, Cristina Romero, Sertan Kaymak, Simena Dinas, Markus Brenner, Xavier Sevillano, Vijay Kumar, Daria Štefić, Konstantinos Bozas, Petar Palasek, Anıl Aksay, Navid Hajimirza, Vangelis Sariyanidi, Karthik Vaiapury, Patrycia Barros, Yixian Liu, Juan Carlos Caicedo and all the other members and visitors of MMV. I am particularly grateful to Fiona, Julie, Virginia, Naeem, Michele and Saverio for helping me proof-read my thesis.

I would also like to thank all my friends from Queen Mary and the other members of QMUL Italian Society.

Infine, vorrei ringraziare i miei genitori, Ornella e Giuliano, per avermi sempre incoraggiato durante i miei studi.

Last, but not least, I would like to thank my parents Ornella and Giuliano for their encouragement throughout all my studies.

Abstract

The objective of the research work presented in this thesis is to study scalable video transmission over peer-to-peer networks. In particular, we analyse how a credit-based approach and exploitation of social networking features can play a significant role in the design of such systems. Peer-to-peer systems are nowadays a valid alternative to the traditional client-server architecture for the distribution of multimedia content, as they transfer the workload from the service provider to the final user, with a subsequent reduction of management costs for the former. On the other hand, scalable video coding helps in dealing with network heterogeneity, since the content can be tailored to the characteristics or resources of the peers. First of all, we present a study that evaluates subjective video quality perceived by the final user under different transmission scenarios. We also propose a video chunk selection algorithm that maximises received video quality under different network conditions. Furthermore, challenges in building reliable peer-to-peer systems for multimedia streaming include optimisation of resource allocation and design mechanisms based on rewards and punishments that provide incentives for users to share their own resources. Our solution relies on a credit-based architecture, where peers do not interact with users that have proven to be malicious in the past. Finally, if peers are allowed to build a social network of trusted users, they can share the local information they have about the network and have a more complete understanding of the type of users they are interacting with. Therefore, in addition to a local credit, a social credit or social reputation is introduced. This thesis concludes with an overview of future developments of this research work.

Contents

List of Tables

List of Figures

List of Abbreviations

4CIF	Spatial resolution, corresponding to 704×576 pixels (see CIF)
ANOVA	ANalysys Of VAriance
BiToS	BitTorrent Streaming
CGS	Coarse-Grain quality Scalability
CIF	Common Intermediate Format, spatial resolution corresponding to 352×288 pixels
DASH	Dynamic Adaptive Streaming over HTTP
DCT	Discrete Cosine Transform
FGS	Fine-Grain quality Scalability
fps	Frames per second
FR	Free-rider
G2G	Give-to-Get
HD	High Definition, spatial resolution corresponding to 1280×720 or 1920×1080 pixels
HEVC	High Efficiency Video Coding
HVS	Human Visual System
LC	Layer Chunk
LSB	Least Significant Bit
MCTF	Motion-Compensated Temporal Filtering
MGS	Medium-Grain quality Scalability
MOS	Mean Opinion Score
MSB	Most Significant Bit

MSE ..... Mean Squared Error

NAL ..... Network Abstraction Layer

P2P ..... Peer-to-Peer

PBE ..... Perfect Bayesian Equilibrium

PSNR ..... Peak Signal-to-Noise Ratio

QCIF ..... Spatial resolution, corresponding to 176×144 pixels (see CIF)

QoE ..... Quality of Experience

QoS ..... Quality of Service

SAQ ..... Successive Approximation Quantisation

SN ..... Social Network

SNR ..... Signal-to-Noise Ratio

SPPM ..... Stanford Peer-to-Peer Multicast

ST ..... Spatial-Temporal (decomposition)

SVC ..... Scalable Video Coding

T4T ..... Tit-for-Tat

TCP ..... Transport Control Protocol

TTL ..... Time-To-Live

UDP ..... User Datagram Protocol

VCL ..... Video Coding Layer

VoD ..... Video on Demand

WSVC ..... Wavelet-based Scalable Video Coding

Chapter 1

Introduction

1.1 Motivation

In the last few years, distribution of multimedia content over the Internet has become very popular due to widespread diffusion of high-speed connections. The most commonly used architecture for these applications is the traditional client-server scheme. This model implies the presence of a client, which is interested in a certain content, and a server, which stores and distributes it. Servers usually deal with several requests at a time and data is replicated to increase availability and achieve fault tolerance. However, this comes with additional costs for the service provider. Moreover, this architecture intrinsically limits the number of simultaneous users, due to its poor scalability. This can be a critical aspect in bandwidth-intensive applications such as video streaming.

Another important characteristic is that, in multimedia applications, video contents of different fidelities are required according to the desired application, such as high quality material for storage and future editing and lower bit-rate content for distribution. In traditional video communications, video is usually processed offline. Compression and storage are tailored to the targeted users according to the available bandwidth and potential receiver or display characteristics. However, this process requires either transcoding of compressed content or storage of several different versions of the encoded video.

Peer-to-peer (P2P) architectures promise to be a scalable and economical alternative to the traditional client-server model, as users contribute by making their own resources

available to the system. Other interesting features of these networks are self-organisation and network path redundancy. However, they present a number of open issues. First of all, some users of the system might behave as *free-riders*. This means that they use resources from other peers without giving anything in exchange. In modern P2P systems, like BitTorrent [1], this problem is tackled by providing peers with incentives to share their upload capacities. Nevertheless, even in this case this behaviour might sometimes pay [2]. Second, data is vulnerable to malicious *pollution* attacks [3]. In this case, a peer deliberately decides to tamper with the content that is being shared, sending other peers corrupt data, which might result in a useless received file (e.g. a video sequence). Finally, users have different upload capacities and P2P networks are highly heterogeneous. All the issues previously introduced are a common feature of many P2P systems, not necessarily performing scalable transmission. In addition, video streaming over P2P networks presents further challenges. For example, data chunks have strict deadlines, represented by their playback time. In fact, if a video part is not received in time, the system may either be paused or part of the video will be skipped. Therefore, this is a challenge that needs to be specifically tackled in the design of these systems.

The problem of network heterogeneity can be partially solved by using Scalable Video Coding (SVC). This allows adaptation of the content to different users requirements by selecting an appropriate sub-set of the original sequence for download and discarding the rest. These codecs can use DCT-like transforms – where DCT represents the Discrete Cosine Transform [4] – such as the standard H.264/SVC [5], or the wavelet transform [6], like the wavelet-based SVC developed at our institution (WSVC) [7]. In both cases, adaptation can be performed in terms of frame size, frame rate or Signal-to-Noise Ratio (SNR), the latter also being known as quality scalability. In this research work, our proposed algorithms have been designed for WSVC. The choice of this codec is relevant from a research point of view, as in a significant number of cases its performance is comparable with or better than the standard. Moreover, WSVC presents appealing features such as efficient management of Fine-Grain (quality) Scalability (FGS). Finally, due to the fact that the two codecs present a similar bit-stream structure, we consider our results obtained in a P2P environment valid for the general SVC case.

Resource reciprocation in P2P networks has been an important area of research for many years. In fact, P2P is based on the idea that users should contribute their band-

width voluntarily. This implies that if there are no rewards for cooperative behaviour and punishments for free-riders a peer that behaves rationally [8] would never share its resources with the rest of the system. Several models, like the original BitTorrent [1] focus on interactions with a limited set of peers over a long period of time and this approach might not be suitable for highly dynamic scenarios, typical of video related applications. In fact, video chunks in P2P video transmission have very strict deadlines and the set of video chunks users are interested in is usually limited and changes repeatedly. On the other hand, this issue may be overcome by requesting video chunks to a larger set of peers, without expecting instant reciprocation of resources. Finally, many alternative approaches see P2P networks as a market, where rules – such as a maximum acceptable credit or debit – can be set in order to create a fair and stable system, where users have incentives to cooperate.

Due to the strict deadlines previously introduced, efficient resource allocation is another typical issue of these systems. In fact, cooperative users might occasionally not be able to to reciprocate resources because they do not have anything to share. If this behaviour repeats itself over time, these users will be considered as free-riders by their neighbours and will receive even less data, creating a vicious circle. As the designer should be able to distinguish between intentional and accidental misbehaving, this problem needs to be tackled within the context of free-riding detection.

Finally, in most systems where there is no global user reputation, decisions taken by single peers only depend on their own knowledge of the network, for example a neighbour's past behaviour. On the other hand, if there are some friendship or "trust" relationships among users, they can share information and better identify misbehaving peers. In the last few years the interest for social networks has grown exponentially. They are most widely known and used as social networking sites, however, their features can be found in other systems, such as P2P networks [9]. In these systems, each user has a profile, which should correspond to a real identity, and establishes relationships with other real users (such as friendship). If a peer considers this information reliable, it can use it to have a better picture of the behaviour of its non-friends neighbours and better distinguish between free-riders and users which have not reciprocated this peer's resources but have been cooperative with its friends. This approach can be vulnerable to *Sybil attacks* [10], where malicious peers can generate multiple sets of identities and

fictitious contributions between them, however, examples of systems that are resistant with respect to these types of attacks have already been proposed in literature [11].

In summary, in this section we highlighted the main issues related to multimedia content distribution using both client-server and P2P architectures. A possible remark is that using a client-server approach it might be much easier to simply replicate the data and increase the number of servers. However, as we have already mentioned, this implies a multiplication of set-up and maintenance costs. The key motivation behind our work is therefore that a P2P approach is much more economical than the traditional one, as it makes use of resources that would otherwise be idle.

1.2 Objectives

Given the context introduced in the previous section, the objective of our research is to investigate the problem of scalable video transmission over a social P2P network. In order to achieve that, we identified the following objectives:

- To perform a subjective video quality evaluation for SVC transmission under different network conditions, to help with the subsequent design of P2P systems that can handle such sequences.
- To propose algorithms for appropriate chunk and neighbour selection, specifically designed for scalable video sequences.
- To investigate the problem of fairness in P2P networks. This is strictly related to tackling the problem of free-riding, by designing specific policies based on peers' contributions that users should obey, or be cut out of the system. Contribution in this context needs to be seen in terms of debit or credit towards other peers.
- To analyse how a social-based approach to this problem (e.g. a "social P2P network" [9]) can improve the resilience of our algorithms, in particular what type of information is useful and how it can be used.
- To research how to create P2P overlay networks that allow to exploit these social-based features.

- To learn how to improve the overall utilisation of the network, which is related to efficiently using the resources of the single peers.

Therefore, the aim of our work is to investigate new subjective quality-aware and credit-based P2P models, which present social networking features, in order to achieve a system where peers fully utilise their upload capacities and free-riding behaviour is punished. Exchanged data will be mainly scalable video sequences.

1.3 Summary of Achievements

In this section, we illustrate the main achievements obtained in this thesis. The starting points of our research work were the original BitTorrent protocol and – for the reasons previously described – a custom, wavelet-based scalable video codec (WSVC), which was developed at our institution. In particular, we focussed on issues related to providing final users with the best possible video quality, free-riding – with and without considering the benefits of exploiting social-based features – and network resources optimisation. The main results presented in this document can be summarised as follows:

- We performed subjective tests to assess perceived quality in the context of scalable video transmission over P2P under different scenarios, for both WSVC and the standard H.264/SVC. These results were compared with objective measurements, which showed a similar behaviour. Our subjective tests, which will be found in Section 3.3, suggested that chunk policies requests for P2P video transmission should aim at providing the final user with a quality that is as constant as possible, discouraging variations for WSVC as well as the standard. Moreover, our test subjects preferred to receive a higher quality at the beginning of the video, as if they were biased by a "first impression" factor, which suggests to increase for example the pre-buffering time of the sequence. Finally, our tests showed that the behaviour of WSVC and the standard SVC are comparable, which indicated that our results obtained with the wavelet-based codec are also valid for H.264/SVC.

- We designed a piece picking policy that allowed to adapt the video bit-rate received by the final user to the available resources of the system, keeping in mind the results obtained in our subjective video quality evaluation. We achieved that

by using a sliding window and prioritising quality layers depending on their importance; the base layer of the sequence (whose role will be explained in Section 2.3.1) is the most critical sub-set of the bit-stream, therefore it was always downloaded before anything else, and the other enhancement layers followed. Adaptation was performed by shifting the window when the playback time approached. Experimental evaluation under fluctuating download bandwidth, which will be illustrated in Section 4.6.2, showed the benefits of using a scalable codec and that changes in the chunks receiving rate corresponded to actual variations in the bit-rate of the received video, which was comparable with the download bandwidth.

- We proposed a neighbour selection policy that only requests the base layer of the sequence to the best peers in the network, in order to reduce the risk that this part of the bit-stream was requested to slow peers and therefore not delivered in time. The best peers in the network are defined as the users with the highest download bandwidths, which alone can provide a rate that is sufficient to receive the base layer in time. Our results, which will be explained in Section 4.6.3, showed that activating and de-activating this policy had an impact in the number of pauses caused by base layer chunks not being received in time, with a significant reduction when the policy was active.

- We studied a simplified version of the problem of video transmission over P2P, without considering many of the non-ideal aspects of a real system. We provided an analytical solution to this reduced problem, which cannot be applied to a real system, but provided ideas that can be applied to more complex cases. In particular, it highlighted the problems related to having "deprived" peers in the network, which have no data to share. As we will discuss in Section 5.2.1, we found that this is an issue that needs to be specifically addressed in a rapidly changing context like video transmission.

- We studied the problem of joint network resource optimisation and free-riding identification. In particular, we proposed a credit-based framework whose aim was to promote cooperation among users – and achieve fairness by cutting out free-riders – as well as providing more data to those users that had less, reducing their risk of being marked as malicious. Our algorithm used a credit line mechanism, which was the maximum accepted difference between uploaded and downloaded chunks

before marking a peer as non-cooperative. Moreover, resource allocation was based on two factors:

  - *credit factor*, which indicated the current debt or credit towards a peer. Giving more importance to this factor resulted in repaying a peer's debt more promptly.

  - *deprivation factor*, which depended on the potential contribution towards a peer. A higher weight assigned to this factor encouraged a more equal chunk distribution.

  Our simulations, which will be found in Section 5.6, showed that free-riders were effectively cut out of the system. Moreover, this approach resulted in a higher received video bit-rate with respect to those cases when optimal resource allocation and free-riding detection were considered as separate problems.

- We extended our previous approach by considering social-based features. We relaxed the constraints about the maximum acceptable credit, and introduced the concept of social reputation, which depended on the information friend peers reported about another user in the network. A peer was considered a free-rider if it either had a low credit or a bad social reputation. Our experiments, whose outcome will be illustrated in Section 6.4, indicated that peers in the social network could benefit from the information from their friends by receiving an overall higher video quality, and that our considerations about credit and deprivation factors were still valid in this context.

- It is intuitive that if peers are connected to random users, friends have low chances to interact with the same neighbours, especially if the network is big. As we will see in Section 6.3.4, this is relevant, as our social reputation is based on the assumption that friend peers can share information about peers they have both interacted with. In order to make the impact of our approach significant, we modified the tracker protocol to connect peers in the social network to friend users, and other potentially uncooperative users to group of friends, which can share information about them.

1.4 List of Publications

Most of the contributions presented in this thesis have been published in international journals, proceedings of international conferences and workshops [12–17]. The detailed list of publications is given below, together with another publication of ours whose content is not described in this document [18], but whose field will be investigated for future work.

Journals

[12] N. Ramzan, E. Quacchio, T. Zgaljic, **S. Asioli**, L. Celetto, E. Izquierdo, and F. Rovati, "Peer-to-peer streaming of scalable video in future Internet applications," *IEEE Communications Magazine, Special Issue on Future Media Internet*, vol. 49, pp. 128–135, March 2011.

[13] **S. Asioli**, N. Ramzan, and E. Izquierdo, "A game theoretic approach to minimum-delay scalable video transmission over P2P," *Signal Processing: Image Communication*, vol. 27, no. 5, pp. 513–521, 2012.

Conferences and Workshops

[14] **S. Asioli**, N. Ramzan, and E. Izquierdo, "Efficient scalable video streaming over P2P network," in *Proceedings of the 1st International ICST Conference on User Centric Media (UCMedia)*, December 2009.

[15] **S. Asioli**, N. Ramzan, and E. Izquierdo, "A novel technique for efficient peer-to-peer scalable video transmission," in *Proceedings of the 2010 European Signal Processing Conference (EUSIPCO)*, August 2010. Also presented at the *Streaming Day 2010 (STDAY) Workshop*, September 2010.

[16] **S. Asioli**, N. Ramzan, and E. Izquierdo, "A game theoretic framework for optimal resource allocation in P2P scalable video streaming," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2293–2296, IEEE, 2012. Also presented at the *Streaming Day 2012 (STDAY) Workshop*, October 2012.

[17] **S. Asioli**, N. Ramzan, and E. Izquierdo, "Exploiting social relationships for free-riders detection in minimum-delay P2P scalable video Streaming," in *Proceedings of the 2012 IEEE International Conference on Image Processing (ICIP)*, pp. 2257–2260, IEEE, 2012.

Other Publications

[18] R. Mekuria, M. Sanna, **S. Asioli**, E. Izquierdo, D. Bulterman, and P. Cesar, "A 3D tele-immersion system based on live captured mesh geometry," in *Proceedings of the ACM Multimedia Systems conference (ACM MMSys)*, 2013.

1.5 Structure of the Thesis

The remaining chapters of this thesis are organised as follows:

Chapter 2 provides a background for our research work. More specifically, it first introduces a few concepts related to subjective video quality and Quality of Experience (QoE), BitTorrent and other P2P systems, with a particular focus on credit-based architectures. Second, it discusses the properties of SVC and describes the main characteristics of WSVC and H.264/SVC, used in our experimental evaluations. Finally, it gives an overview about social networks, explaining how it is possible to generate graphs that capture their fundamental properties.

Chapter 3 describes our subjective evaluation of scalable video transmission over P2P. First of all, we introduce the problem of video quality assessment, the objective metrics used in this field and present a few related studies. Second, we clarify the methodology used in our subjective evaluation, our testing environment and scenarios. Third, we show the outcome of both our objective and subjective evaluations. Finally, we compare these results and draw conclusions that will be useful for the design of our systems.

Chapter 4 begins with a description of a block diagram representing a general architecture for a social P2P system for scalable video transmission. Subsequently, we introduce the problem of efficient chunk requesting in scalable video transmission over P2P. After providing the related background, we describe our piece picking

and neighbour selection policies and we shortly discuss their implementation. Our experimental evaluation shows the impact of our proposed approaches on the behaviour of our system. Finally, we critically discuss the positive and negative aspects of this technique.

Chapter 5 proposes an algorithm for joint free-riders detection and network resource optimisation. We start by introducing the problem, then we discuss a simplified version of the problem of video transmission over P2P and we propose an analytical solution. Related systems that tackle free-riding and techniques that aim at optimal resource allocation are then described, before introducing our approach, which is based on some of the conclusions arising from the solution to the simplified problem. A credit-based mechanism is then explained. Its aim is to identify and cut out free-riders, while at the same time our resource allocation mechanism tries to find the best trade-off between promptly repaying a peer's "debts" and uploading data to deprived peers. The chapter continues with some considerations about the implementation of this architecture using a network simulator, our tests and an analysis of the pros and cons of this solution.

Chapter 6 adds social networking features to the approach described in Chapter 5. After introducing the possible benefits deriving from the use of social-based features – such as the possibility to join forces when trying to identify free-riders – we describe a few techniques that consider P2P networks as social networks or use a social reputation. Subsequently, we explain our algorithm to compute a peer's social reputation based on the interactions it had with another peer's set of friends. As this approach would be ineffective without a ponderate construction of the overlay network, we also introduce a new tracker protocol that takes social relationships among peers into account. After showing the results of our experiments, we discuss the strongest and weakest aspects of this solution.

Chapter 7 concludes this thesis, summarising our main contributions. Finally, we present a few possible directions for our future research work. In particular, we will focus on more game theory-oriented approaches, briefly introducing a new technique based on Bayesian game theory.

Chapter 2

Background

The chapter provides some background about the context in which we developed our research work. This is a general description, as more specific techniques will be explained in the "Related Work" section of each chapter that explains one of our proposed architectures. The rest of the chapter is organised as follows: Section 2.1 provides an introduction to QoE and how it relates to subjective video quality; Section 2.2 gives a general definition of P2P and discusses the history of such systems before analysing the main concepts of BitTorrent in detail and describing a few credit-based systems; Section 2.3 discusses the advantages of a scalable approach to video streaming, provides a general architecture for SVC systems and briefly illustrates the two codecs used for our experiments: WSVC and H.264/SVC; Section 2.4 introduces a few concepts related to social networking and defines the key properties of such networks; finally, Section 2.5 concludes the chapter.

2.1 Subjective Video Quality and Quality of Experience

According to a quite recent definition by Crespi et al. [19], in the context of multimedia applications QoE "represents the human centric quality aspects, unlike QoS which is merely a technology centric approach. [It] is a blue print of all human needs, desires and perceptions concerning a service and/or product in a specific context."

This definition clearly states that the key aspect of QoE is that it is not necessarily based on objective metrics, like Quality of Service (QoS), but on how the final user *perceives* a certain content. Defining this quantity in the context of video transmission over the Internet is itself a challenging task [20]. In fact, quality metrics need to consider the application a user is interested in, as well as the impact of delivery to the final user itself. In particular, they should capture how playback interruptions, received bit-rate, frame rate, changes in the bit-rate and pre-buffering times affect the final user experience. Moreover, another key aspect of Internet video is user engagement, which can be represented for example by the percentage of a video watched by a user before switching to something else, and which does not necessarily relate to objective metrics that quantify how much an encoded video differs from the original sequence.

In their study, Balachandran et al. identified three key factors that explain why assessing video QoE is an arduous challenge [20]:

- **Complex Relationships**: The relationship between objective metrics or parameters and the user experience may, under many circumstances, not be linear. For example, higher received video bit-rates do not always result in a better user experiences, especially when they result in a high number of quality switches. Another example showing the complexity of the problem is given by the different types of experience users might be interested in. In fact, a few subjects that were following live events preferred to stream the video in the background while performing other tasks and only occasionally watching it. In this case, their QoE was positively affected by a lower received frame rate, as it resulted in a smaller computational power needed for decoding, while the remaining computing resources could be used for their other tasks.

- **Metric Dependencies**: Different parameters are often linked and improving one aspect of the video playback might affect another one negatively. For instance, performing bit-rate adaptation to the current download bandwidth allows to reduce the number of pauses or buffering, while repeated bit-rate changing might be evaluated very negatively.

- **Impact of Content**: The content itself may influence the judgement of the video quality assessors. The two main aspects considered in the study are whether the application is live streaming or VoD, and the user's interest in the content itself.

The debate regarding these challenges is still open in the scientific community and a function that accurately relates QoE or user engagement with other objective metrics has not been accepted as standard yet. In our study in Chapter 3, we will focus on analysing perceived video quality under different network scenarios. As in our work we will be more focussed on measuring opinion scores of video quality rather than user engagement, we prefer to refer to our study as subjective video quality evaluation instead of QoE assessment.

2.2 P2P Systems

P2P networking architectures allow a range of new applications that can exploit distributed storage and parallel usage of computing resources. The following definition of P2P [21] was given in 2001, but it is still valid nowadays:

> *A distributed network architecture may be called a Peer-to-Peer [...] network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers,...). These shared resources are necessary to provide the Service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (Service and content) providers as well as resource (Service and content) requestors (Servent-concept).*

This is a very general definition and also includes systems like SETI@home [22] or Skype [23]. In the first case, the shared resource is computational power, while in the second it is network bandwidth. However, from now on we will only focus on P2P systems used for file sharing.

2.2.1 A Brief History of P2P

The first popular P2P client was Napster [24], developed in 1999. It was a *centralised* system, since its operation was based on a central server that kept track of all users and available resources in the network. One year later, in 2000, Gnutella [24] was developed.

The original system was purely *decentralised*, as nodes were directly connected to each other, without the need of any central entity. This made the system robust, since failure of any (single) node did not significantly affect the overall performance of the system. However, Gnutella was an *unstructured* system, which means that there was no strict control over network topology. As a matter of fact, in this approach it was not possible to provide a directory service and the client itself made massive usage of flooding. As opposite to unstructured ones, *structured* systems [25] have been designed over the past few years. In these networks, nodes and data are placed according to certain rules, with the aim of locating files in an efficient way. Queries are quick and efficient thanks to the use of distributed routing tables. Comparison with unstructured systems has proven structured approach to be more scalable, more reliable and fault tolerant. In 2001, creators of KaZaA [26] proposed an *hybrid* solution, half-way between centralised and decentralised systems. In this system, peers are divided into two categories: regular and super nodes. The latter have more capacity than the average nodes and are chosen to perform additional functionalities like keeping track of peers and data in a certain portion of the network and keeping a list of other super nodes. Queries are only performed through these special peers, using therefore "intelligent" flooding. A significant improvement is represented by BitTorrent, which deserves a special mention, as it is the starting point of all our proposed techniques.

2.2.2 BitTorrent

BitTorrent is a game theory-based P2P protocol developed by Bram Cohen in 2003 [1] and the main idea behind it is that users' download rates should be proportional to their upload rates, in order to provide a fair mechanism and motivate users to share more. Free-riding is occasionally accepted, but only if there is enough spare capacity in the system. Ever since it was designed, several major improvements have been added, in order to eliminate single points of failure and add new features like Video on Demand (VoD) or live video streaming. BitTorrent protocol is thoroughly described in this chapter, as it will be the starting point of our architectures described in Chapter 4, 5 and 6.

Base Working Principles

If a user wants to publish one or more files in BitTorrent, it should create a *torrent* file. It contains information such as size, name of these files, some checksums and the address of a *tracker*, which is a server that helps peers interested in a certain content find each other. Moreover, files are split into several pieces of a fixed size, which can be typically picked from a 32 kilobytes to 2 megabytes range. Each piece is associated with an ID and a SHA1 hash, which is written in the torrent in order to check integrity of the piece once it has been downloaded – and tackle the issue of pollution. BitTorrent transfers data using the Transport Control Protocol (TCP) and, to improve overall performance of the system and avoid delays, each peer should have an optimum number of requests pending at the same time. Pieces are cut into sub-pieces and typically five requests are pipelined at once. When a new sub-piece is received, another request is sent. The user that publishes a file becomes a *seeder* for that particular file and other peers, which are called *leechers* in this case, can download from it. A peer interested in that file should download the associated `.torrent` file and contact the tracker, which provides it with a partial list of users that own all or part of it. When a leecher completes its download, it becomes a seeder and the set of all seeders and leechers is called a *swarm*. An important remark is that in general BitTorrent clients will not allow users to search for torrents, which can be located on ordinary web servers or can be exchanged in many other ways.

Piece Picking

In the previous section, division of pieces into sub-pieces was mentioned. The general rule is that a piece should be completed before requesting sub-pieces belonging to other chunks, in order to have complete pieces as soon as possible. In fact, pieces will not be shared until their download is terminated. This policy is called *strict priority* and it is the main rule that any BitTorrent client should obey. Under these circumstances, the main issue becomes choosing which new piece to pick for downloading. All BitTorrent clients keep track of which pieces are owned by their connected neighbours and sort them according to number of occurrences; in this context rarest pieces are considered more interesting and are chosen first. This behaviour has many advantages:

- If the original seeder for a specific content leaves the P2P network after a short time, it is more likely that at least one complete copy is transmitted – even if to different users – using this technique, instead of selecting pieces at random. Therefore, the full content remains available to the users even after the original peer has left.
- Similarly, if a piece is owned by only a few peers, it will be picked first, since the probability that these peers disconnect and therefore the piece is lost is quite high.
- If a piece is extremely rare, other users might be interested in it, which means that other peers might download it once it is owned by a peer. Since download bandwidth is proportional to upload bandwidth, owning rare pieces will grant higher download rates from other peers.

An exception to this rule is represented by the first piece, which is chosen at random. Since this piece is downloaded without giving anything in exchange, it is important that it is downloaded as soon as possible, in order to have something to share. In this case downloading a rare piece is not convenient, since it might be owned by just one peer and transfer could be slow. Therefore, pieces are randomly selected until one complete piece is received. After that, the policy switches to rarest first.

In the final part of the download, when all the missing sub-pieces are being requested, the protocol enters the so-called *endgame mode*. This should be a very short period, in order to avoid wasting resources, and it consists of requesting all these missing sub-pieces from all the neighbours. When a sub-piece arrives, all the other requests for it are cancelled. The reason for this is that, if a peer downloads its last sub-pieces from another peer which has very slow transfer rates, it has to wait for an unnecessarily long time. With this technique, instead, downloads are completed very quickly, while usage of extra resources is minimal.

Choking and Unchoking Algorithms

Since in BitTorrent there is no such thing as central resource allocation, each peer has to work independently to maximise its own performance. What happens in practice is that each peer tries to download from all the peers that will allow them to, while *choking* or *unchoking* (uploading to or refusing to upload to) other peers. All the algorithms that can be used should try to reach Pareto efficiency [8]. In computer science, this

is an algorithm that tries to reach global optimum by maximising the profit of every single peer. Since peers will mostly upload to peers that are "generous" towards them, this algorithm is called *tit-for-tat* (T4T). In practice, each peer has a fixed number of unchoked neighbours, typically four, and the problem that the algorithm tries to solve is deciding which peers to unchoke. The default behaviour is uploading to peers that give something in exchange, but if there are some free connections they are used to unchoke random peers, to "explore the network" and try to find better peers. If all the connections are used, instead, all peers will be sorted and unchoked according to their upload rates, except for one, which will be *optimistically unchoked*. This means that one peer is unchoked regardless of its upload rate. An exception to this rule is represented by BitTorrent's anti-snubbing policy. In fact, if a peer does not receive a piece from another peer in over a minute, it assumes it is being snubbed by the other peer and will refuse to unchoke it, except by optimistic unchoking. Most of the times this strategy leads to more than one peer being optimistically unchoked and helps a peer find better neighbours more quickly.

When a leecher becomes a seeder, it is of course no longer interested in earning a profit from any exchange and it simply uploads to the peers that have higher transfer rates, in order to maximise the download rate of the entire system. The assumption behind this strategy is that high download rates will imply high upload rates to other users. A potential limitation is that, however, not having a feedback from these users, seeders might be uploading data to free-riders. It is also worth mentioning that according to a study by Piatek et al. [27] incentives provided by the BitTorrent protocol are not robust enough, as peers can improve their utility by unilaterally changing their strategy. In this study, a modified BitTorrent client called *BitTyrant* is proposed. The strategy followed by BitTyrant peers differs from the original protocol in terms of size of unchoked peers set and amount of contribution towards them. The modified algorithm will try to find peers that reciprocate resources in exchange of a lower upload bandwidth, and increase the number of unchoked peers until the benefit of an additional unchoked user is outweighed by the lower resource reciprocation probability from other peers.

In the next section, we will analyse a different approach to resource reciprocation, which considers P2P as a market.

2.2.3 Credit-based P2P Systems

The T4T strategy implemented by BitTorrent requires that two peers exchange data in both directions at the same time, and this mechanism works well when the chunks that are being transmitted belong to a very large set. On the other hand, in the context of video transmission peers are interested in the chunks that immediately follow a certain playback position, which represents a much smaller set. Alternative approaches to T4T consider P2P system from a more economic-oriented point of view by introducing credit-based systems. These are systems that model P2P networks in terms of resources that users can trade and peers are associated with a reputation, which depends on their behaviour in the network.

A Barter-based Solution

In 2006, Peserico proposed a market-oriented P2P architecture based on barter [28]. One of the most interesting properties of this system is *wealth storage*. That is, a perishable good can be traded for something that has a value which remains constant over time. An example of goods that lose value quickly in this context is represented by P2P video chunks, as they are of no use once their playback time has passed. In the proposed solution, such resource can be traded in exchange for the promise of future commodities. The main challenge is represented by ensuring that this promise will be kept, however, this can be achieved by offering peers an incentive, such as learning what the other user has gained from the exchange.

For example, let us suppose that there are two peers in the network, A and B, which simultaneously barter one unit of two commodities a and b. Supposing that A would have been willing to trade $1+\epsilon$ units of a in exchange for 1 unit of b and that symmetrical assumptions apply to B, both peers have gained a surplus and have *de facto* opened a credit line with each other. Peer A can therefore now borrow an additional ϵ of b, and B will not experience loss even if the other peer does not give anything in exchange (as B would have been willing to trade $1 + \epsilon$ of b in exchange for 1 unit of a in any case).

Karma and Reputation-based Schemes

Vishnumurthy et al. propose a system based on *karma* [29], which is a parameter analogous to user reputation that depends on how well a peer has behaved towards the rest of the community. Despite it being a global parameter, such solution has been designed for completely decentralised systems. Therefore, karma values are stored by other nodes, also called *Bank nodes* and peers act both as content consumers and guardians for other nodes. Information is heavily replicated to achieve fault tolerance and resistance to collusion attacks. Nevertheless, this system is also vulnerable to phenomena like discrepancy between the karma reported by different nodes due to lack of synchronisation.

We denote with $Bank_A$ the set of nodes responsible for tracking the transactions involving node A and storing A's karma. When A is interested in a specific content from another node B, both nodes need to agree on a price, which could be for example the result of a bidding process. Karma transfers need to be fair and preserve the total amount. Therefore, in this specific example, the karma of A should be reduced of the same amount as the one of B is increased. As this process is vulnerable to tampering, this is achieved through a provable transfer of karma from A to B, followed by another provable transfer of data from B to A. This operation is performed through the aid of $Bank_A$ and $Bank_B$, which supervise the operations through the exchange of messages. When nodes in the bank sets disagree, a majority vote is called.

An important observation the authors make is that peers can be free-riders not only in terms of refusing to upload data to other users, which can overcome by only allowing users with a positive karma to send requests, but also in terms of refusing to act as banking nodes from other peers, while using the services provided by other users. Finally, this system proposes to be a distributed but global reputation mechanism. The reliability of the system comes at the expenses of complexity, and the system is still vulnerable to collusion attacks when a node creates several identities and these peers become its guardians.

On the other hand, the solution proposed by Gupta et al. is a global reputation-based scheme which relies on a *reputation computation agent* [30]. In this system, a peer K can increase its credit and its debit by performing different actions and its reputation depends on different factors:

- **Query-Response Credit**, which is awarded to a peer just by being online and replying to queries from other users.
- **Upload Credit**, proportional to the contribution a user has given to the rest of the network. This value also considers the fraction of its bandwidth a peer is sharing.
- **Sharing Credit**, which depends on the amount of shared content. This quantity has been introduced to give credit to those peers that offer hard to find content and because of this do not get much upload credit.
- **Download Debit**, increased by downloading a file from another user.

A peer's reputation is given by the sum of all its credit components minus its debit component. The authors also propose another model, where the debit component is not considered, but credit expires. Reputation can subsequently be used by other peers as an aid for their decisions regarding resource allocation.

The systems described in this section rely on a global reputation, and need to address the problem of estimating this parameter correctly. Moreover, they represent very generic architectures and have not been specifically designed for video transmission. However, concepts introduced here, like debit and credit, will be used in Chapter 5 and Chapter 6 in the description of our proposed approaches.

2.3 Scalable Video Coding

The most important aspect of SVC is the possibility to encode a sequence once and decode it many times, in many different ways, according to specific application requirements. In order to achieve this, it is necessary to modify the traditional encoding and decoding scheme. Moreover, it is also necessary to define which types of scalability we want to use. First of all, we will introduce some general concepts about SVC, and second we will briefly describe the main characteristics of two codecs we used for our studies; they are based on two different tools: DCT [4] and wavelet transform [6].

2.3.1 SVC Modules

A SVC architecture consists of three modules: encoder, extractor and decoder [31]. The whole scheme is shown in Figure 2.1. The encoder takes an uncompressed YUV sequence as input and produces a scalable bit-stream and its description as output. YUV indicates that each frame of the original sequence is represented by its luminance component (Y) and two crominance components (U and V). The resulting description can be either interleaved within the encoded video sequence or stored in a separate stream, e.g. a text file. The whole sequence has a very high quality, and compression in some cases can even be lossless.

Figure 2.1 shows that different qualities and spatial resolutions can be interleaved in the bit-stream. Light blue data is associated to small spatial resolution, while dark blue bit-stream parts correspond to a higher resolution. These are further divided into smaller parts, that correspond to different qualities. If the extractor completely removes the dark blue parts, the output will be a high-quality sequence with a small spatial resolution. On the other hand, if the extractor truncates both light and dark blue data, this operation will result in a high-resolution video with a low quality. Finally, the green part represents the *base layer* of the sequence, which is the only part that cannot be truncated.

The extractor has in general very low complexity. It produces another scalable sequence, which can be further truncated. Finally, the decoder converts the extracted video back to YUV format, performing operations that are symmetric with respect to the encoder.

2.3.2 Scalability Functionalities

Video extraction, which also takes the name of video adaptation, should be done in a very simple manner, for example parsing the bit-stream and removing the unwanted parts. To achieve that, it is necessary to encode the video in such a way that different bit-stream parts are encoded in a hierarchical fashion with respect to the video parameters we want to scale. It is possible to define three basic types of scalability: spatial, temporal and quality [31]. The latter can also be referred to as SNR (Signal-to-Noise Ratio) scalability. We have spatial scalability when there exists a subset of the original

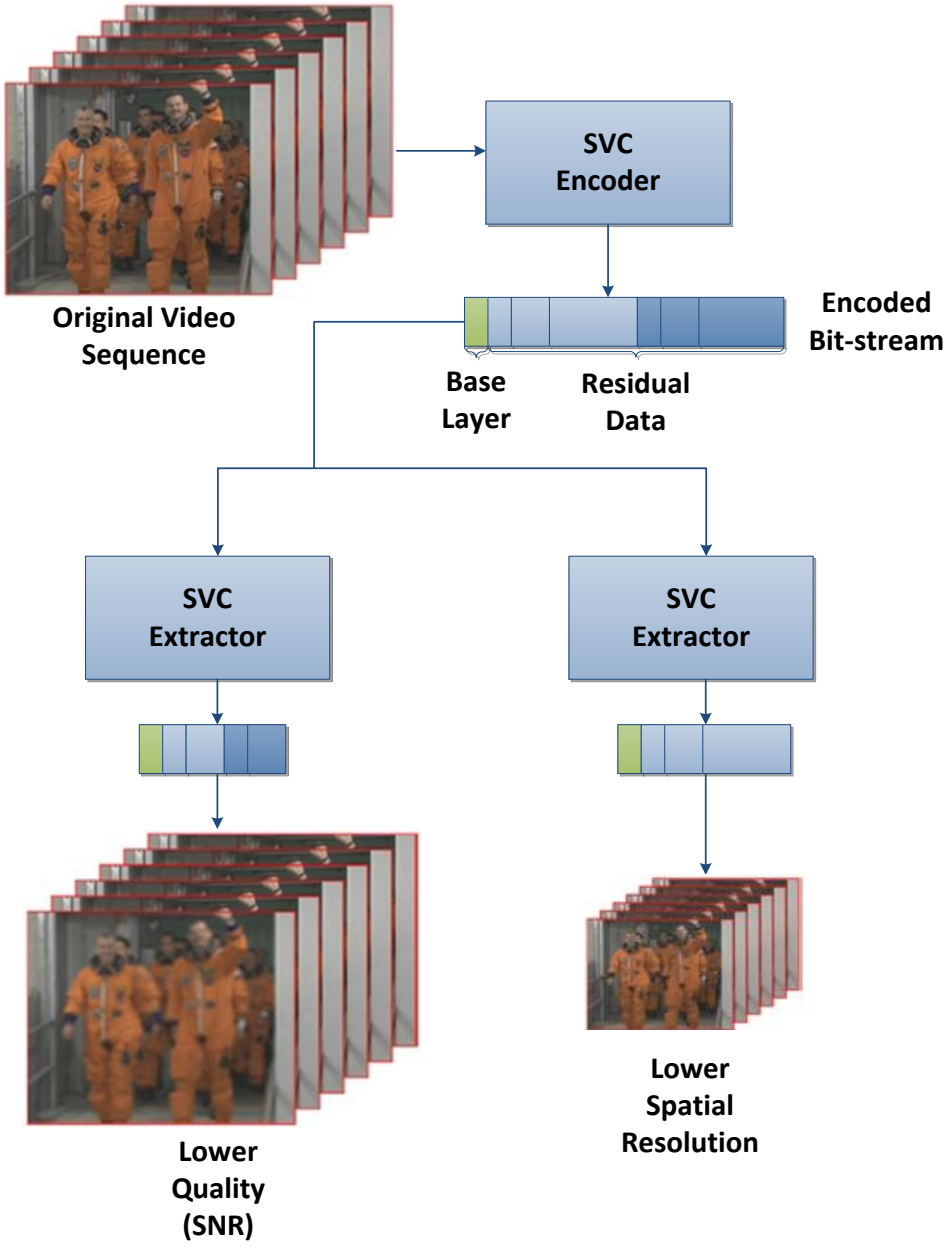


Figure 2.1: Building blocks of a generic SVC system.

sequence that allows to reconstruct the video with a smaller spatial resolution. Temporal scalability holds if a (different) subset corresponds to a lower frame rate version of the sequence. Finally, in order to achieve SNR scalability, different subsets of the video should allow its reconstruction with different qualities. These properties can also be combined; for example, given an original sequence that has 4CIF resolution (704 × 576 pixels, corresponding to four times the CIF – or Common Intermediate Format – resolution) and a rate of 30 frames per second (fps), it could be possible to extract a CIF version of the video (352 × 288 pixels) with 15 fps and a lower quality. An example of the original sequence and scaled sequences is shown in Figure 2.2.

2.3.3 WSVC, a Wavelet-based SVC

Wavelet-based Encoder

Differences between scalable video codec architectures lie in the design of encoding and decoding modules. However, since the decoder simply performs operations that are inverse with respect to the encoder, it will not be described in this section. The scalable video codec used in our proposed approaches is being developed by Multimedia and Vision Research Group (MMV) at Queen Mary University of London since 2006 [32]. Its encoder module can be divided into two blocks: the first one calculates a spatial-temporal transformation of the sequence, while the second one performs the optimal truncation of the bit-stream.

The aim of spatial transforms of images is to compact their energy and de-correlate their content, making their compression simpler and more efficient. A peculiar aspect of the wavelet transform – which this codec is based on – is that it is applied to the whole image at once. In video compression, these concepts are extended to a third dimension (time), in order to exploit temporal correlation between frames.

In WSVC, Spatial-Temporal (ST) decomposition consists of two steps: a motion compensated temporal filtering (MCTF) [33] and a 2D discrete wavelet transform. They provide, respectively, temporal and spatial scalability. These two operations are necessarily performed separately, which allows a degree of freedom. In fact, temporal decomposition can either be performed before the spatial one ($t+2D$ scheme) or after it ($2D+t$ scheme). Decomposition of one level is equivalent to the operation of analysis. The pro-

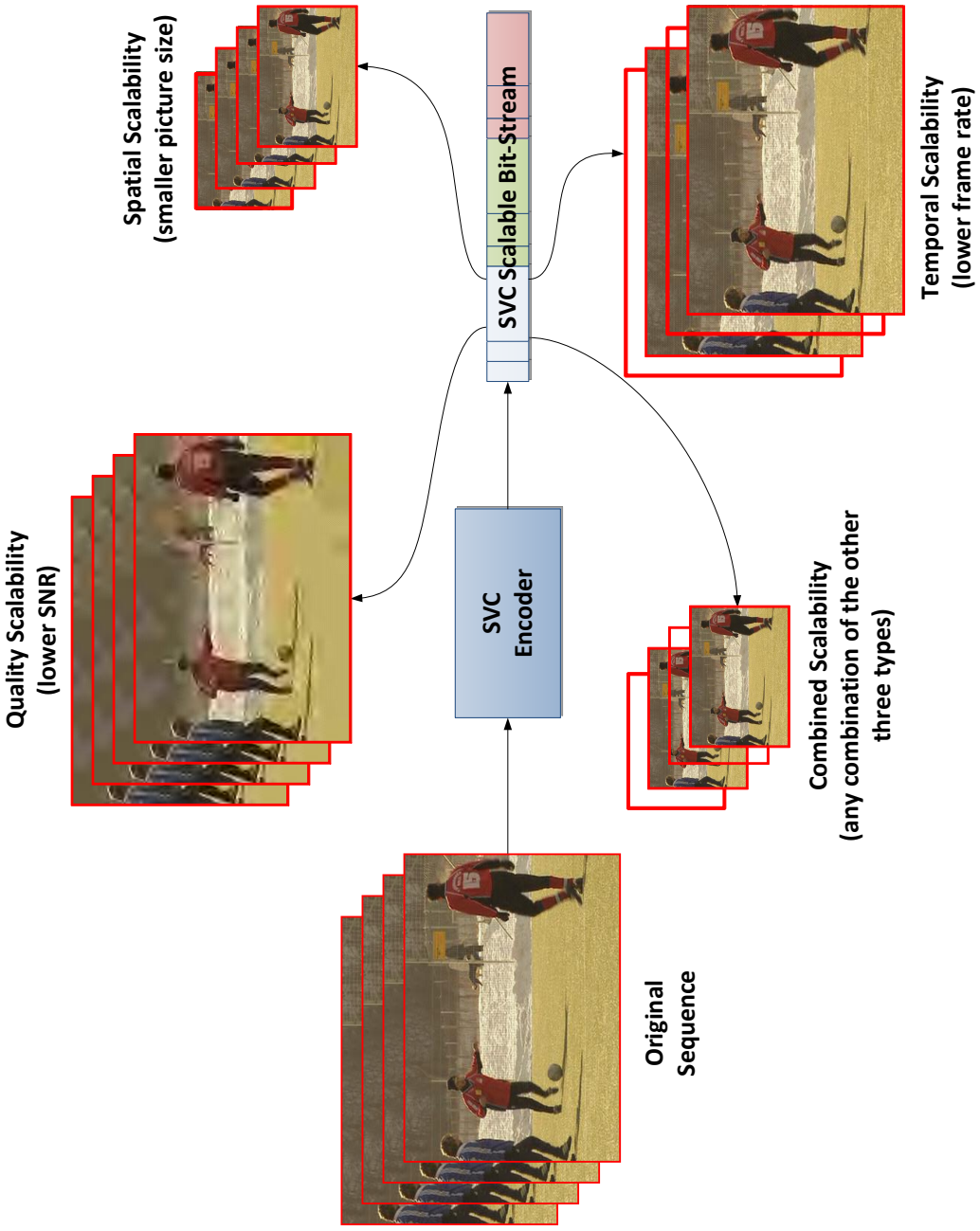


Figure 2.2: Examples of basic and combined scalability.

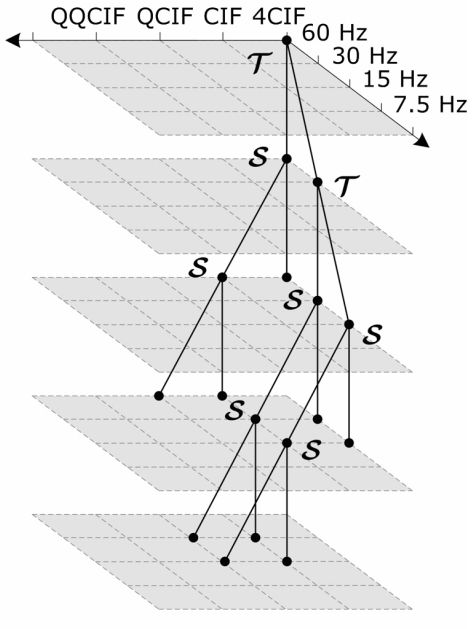


Figure 2.3: A ST decomposition tree for t+2D decomposition [32].

cess reversing the decomposition is reconstruction, where a one level reconstruction is equivalent to the operation of synthesis.

ST decomposition can be represented by an ST tree structure [31,32]. This is shown in Figure 2.3 for two temporal and two spatial levels of transform. Here, T and S denote one step of the temporal and spatial decomposition, respectively. Each level of decomposition creates a node in the ST tree, called an ST node. To perform reconstruction of the video sequence at the decoder's side, the data contained in the leaves of the ST tree has to be transmitted. In the following the leaves of an ST tree will be referred to as ST sub-bands. Each ST sub-band is represented with 2D coordinates corresponding to its temporal (T) and spatial resolution (S), which is denoted as (T, S). By convention, a lower value of the coordinate corresponds to a lower value of the spatial or temporal resolution.

In order to achieve quality scalability, wavelet coefficients are bit-plane [34] encoded. Since the result of a ST wavelet transform is a set of transformed frames, bit-plane encoding can be performed on a frame-to-frame basis by using algorithms developed for scalable coding of still images. This algorithm applies successive approximation quantisation (SAQ) on the wavelet coefficients generated by the 3D wavelet transform in order to encode them progressively, from the most significant bit (MSB) to the least significant

bit (LSB). This operation results in a video sequence that has optimal distortion at the specified spatial and temporal resolutions, given an appropriate set of target bit-rates.

Bit-stream Description and Extraction

Adaptation to lower fidelity content, (i.e. quality, spatial or temporal resolution) can be performed by simply removing those parts of the bit-stream that represent a higher fidelity of the video with respect to the targeted one. For adaptation of the scalable bit-stream, an extractor can be used.

In order to achieve efficient extraction, a WSVC bit-stream consists of data packets called atoms [31]. An atom represents the smallest entity that can be added to or removed from the bit-stream, and groups of atoms form layers. Following such organisation, the extractor simply discards from the bit-stream the group of atoms that are not needed to achieve the desired resolution. For an easier interpretation of the extraction process, the bit-stream can be represented in a 3D Temporal resolution, Spatial resolution, Quality (T,S,Q) space, as shown in Figure 2.4 for example of $tl = sl = ql = 2$. tl, sl, ql are the number of refinement layers in the quality, temporal and spatial domains respectively. With the exception of refinement layers, in each domain there exists a basic layer which is denoted as 0^{th} layer and cannot be removed from the bit-stream. Therefore, in the example shown in Figure 2.4 we have 3 quality, 3 temporal and 3 spatial layers. Each atom has its coordinates in T-S-Q space, which are denoted by (T,S,Q). If (i, j, k) represents the desired quality, temporal and spatial resolution of the adapted video, where $i \in 0, 1, \ldots, tl, j \in 0, 1, \ldots, sl$ and $k \in 0, 1, \ldots, ql$, then, in the extraction process, the atoms with coordinates $T > i, S > j, Q > k$ are simply discarded from the bit-stream. In Figure 2.4, the atoms that are highlighted are the ones remaining in the final bit-stream after the extraction process for which $i = j = k = 1$.

A high-level description of the WSVC bit-stream is illustrated in Figure 2.5. The bit-stream begins with the main header, which contains parameters that define general information about the encoded video. The most notable ones are spatial resolution, frame rate and number of spatial, temporal and quality layers. Then, an optional header follows. It is composed of parameters that define more advanced encoding settings, such as motion block size or type of wavelet transform. The remaining part of the bit-stream is divided into Groups Of Pictures (GOPs) of the encoded video. A GOP is the

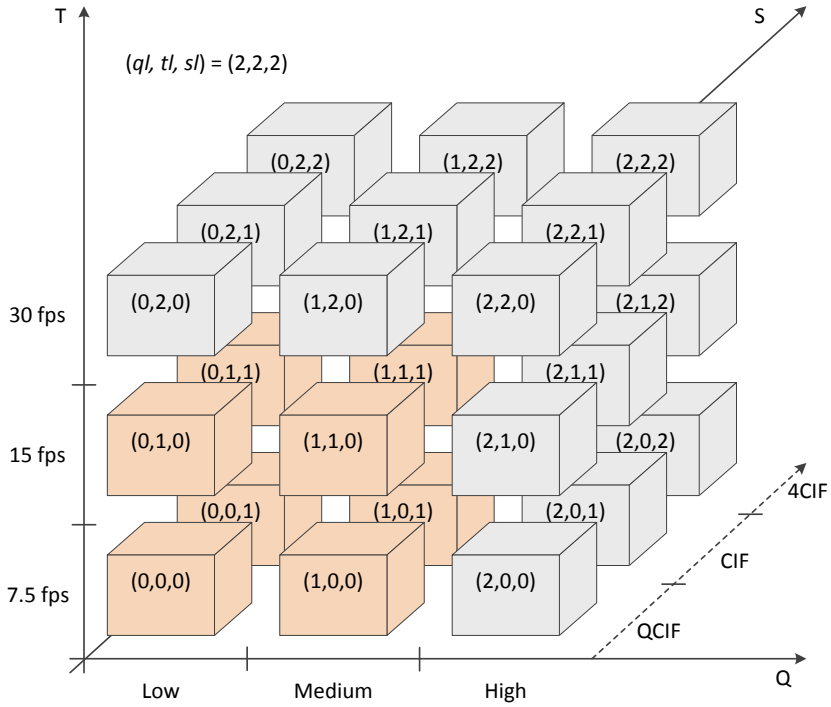


Figure 2.4: 3D representation of a scalable video bit-stream.

basic unit of input video on which the scalability is imposed, and for which a (T,S,Q) representation can be used. Each GOP has a header with parameters that describe some general information about that GOP. The GOP header is followed by its payload, which is divided into atoms. Each atom contains a tag specifying its size, motion vector and texture information. Since the size of the atoms is not constant, the number of bytes that an atom tag occupies in the bit-stream varies as well; its minimum size is 1 byte and the maximum is 4 bytes.

During the extraction process, the WSVC extractor internally creates a (T,S,Q) representation of the bit-stream shown in Figure 2.4. That representation only exists in the memory of the extractor. Along with the (T,S,Q) coordinates of the atoms, it also contains information about the starting byte and length of the particular atom. Therefore, for each atom, the extractor knows on which byte in the bit-stream it starts and how big it is. Based on that information, when an atom has to be discarded, the extractor simply removes the corresponding part of the bit-stream and concatenates the remaining parts with the previous one. An important remark is that WSVC supports fine granularity. In other words, instead of extracting a specific number of quality layers,

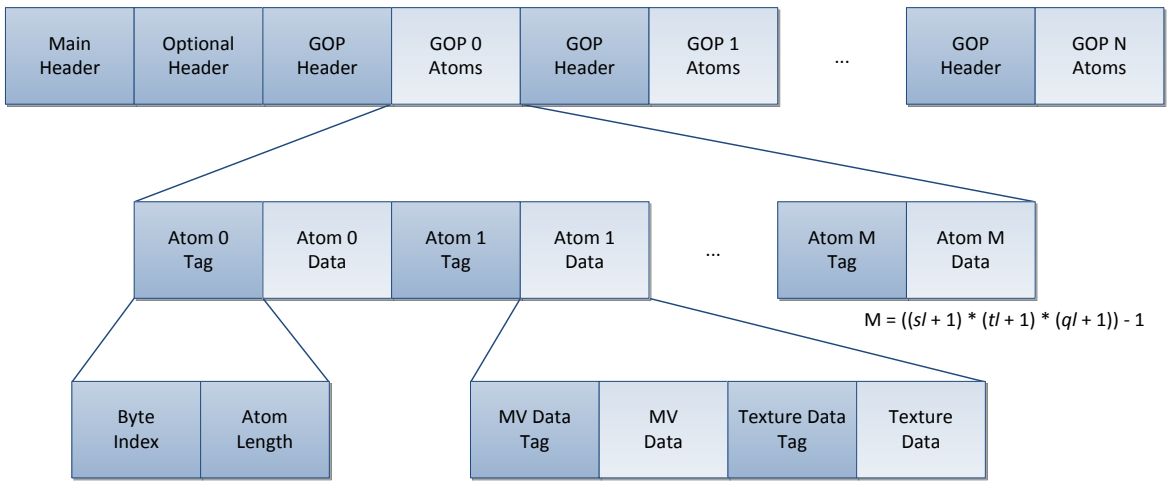


Figure 2.5: High level description of a scalable video bit-stream.

it is possible to specify the bit-rate of the extracted video. However, the SNR of the video might not be optimal for this specific bit-rate, since, as we previously mentioned, it is optimised with respect to some extraction points.

2.3.4 H.264/SVC, the Scalable Extension of H.264/AVC

H.264/SVC [5] has been standardised as the scalable extension of H.264/AVC [35]. The design of these codecs concerns both a *Video Coding Layer* (VCL) and a *Network Abstraction Layer* (NAL). The first one is responsible for the generation of the coded bit-stream, whereas the second formats this data and generates the respective headers, in order to allow exploitation of the functionalities of the codecs. Encoded bit-streams consist of several NAL units, made of a header and a payload. These units can be either VCL NAL – containing coded data such as slices – or non-VCL NAL, the most important of which contains parameter sets. In general, scalability is obtained by discarding the desired NAL units and reconstructing the video with the remaining data.

The VCL follows a *block-based hybrid approach* [35], similarly to other codecs like MPEG-2 [36] and H.263 [37]. The division of frames into smaller coding units follows the partition into *slices* (parts of the frame that do not depend on any other region of the same frame) and *macroblocks*, typically made of 16×16 luma and 8×8 chroma samples (if the sampling format is 4:2:0 [38]). The codec supports three types of slices,

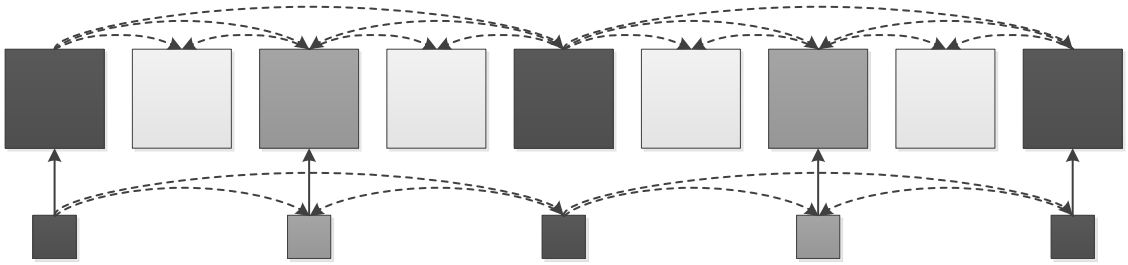


Figure 2.6: Inter and intra-layer prediction dependencies for spatial scalability.

whose difference lies in the information they require for their decoding and the slices they depend on:

- **I-slices**: coding is *intra-picture* only and uses spatial interpolation from neighbouring areas within the same slice.
- **P-slices**: both intra and *inter-picture* coding, where the prediction only depends on one other signal.
- **B-slices**: similar to P-slices, however the prediction is the weighted average of two signals (e.g. a "past" signal and a "future" one).

The residual signal (prediction error) is transformed with a filtering that is very similar to DCT, quantised and finally subjected to entropy coding.

Temporal Scalability

As far as temporal scalability is concerned, it is obtained by imposing constraints on the slices that are used as reference for motion compensation. For example, let t be a temporal layer identifier varying between t_0 and t_N, where t_{i+1} denotes higher temporal resolution with respect to t_i. Predictions of slices belonging to t_i can only be made from those with a temporal index $k < i$. In general these predictions can be generated using more than one reference, as in H.264/AVC, and it is possible to avoid the usage of future slices.

Spatial Scalability

In order to obtain spatial scalability, this codec uses the traditional multi-layer coding approach. That is, in addition to intra-frame prediction and motion compensation inside each layer, inter-layer prediction is also used, as shown in Figure 2.6. For an enhancement layer frame, the reconstructed prediction can be generated by using motion compensation techniques inside the layer, by upsampling a lower layer frame or calculating an average between the upsampled and temporal prediction. The figure also shows how spatial and temporal scalability can be combined.

Quality Scalability

Quality scalability can be seen as a particular case of spatial scalability where two layers q_0 and q_1 have the same frame size and therefore no upsampling is required. For different layers, different quantisation step sizes are also used when encoding the residual. This case is known as *Coarse-Grain quality Scalability* (CGS) and as the name suggests only allows a limited number of decoding points. In order to achieve *Medium-Grain quality Scalability* (MGS) the concept of *key picture* is introduced. An issue that may be encountered when decoding scalable sequences is *drift*. It is the lack of synchronisation between motion-compensated prediction loops at the encoding and decoding sides. For a subset of frames, the key pictures, the motion vectors used to generate the prediction must be the same for all layers. Moreover, a flag is also transmitted, which indicates which frame has been used as reference to generate the prediction. Using this technique allows to have more flexibility on the extraction points, hence the MGS, as it allows to split the information more efficiently in the different NAL units.

2.3.5 Final Remarks about SVC

Both WSVC and H.264/SVC allow a layered representation of the video bit-stream. In fact, it is sufficient that the corresponding atoms or NAL units are grouped according to any of the scalability parameters. Therefore, if a user wants to exploit only one type of scalability, such as quality, it is possible to assume the sequence to be divided into GOPs and quality layers, as shown in Figure 2.7. Layer q_0, which is the base layer

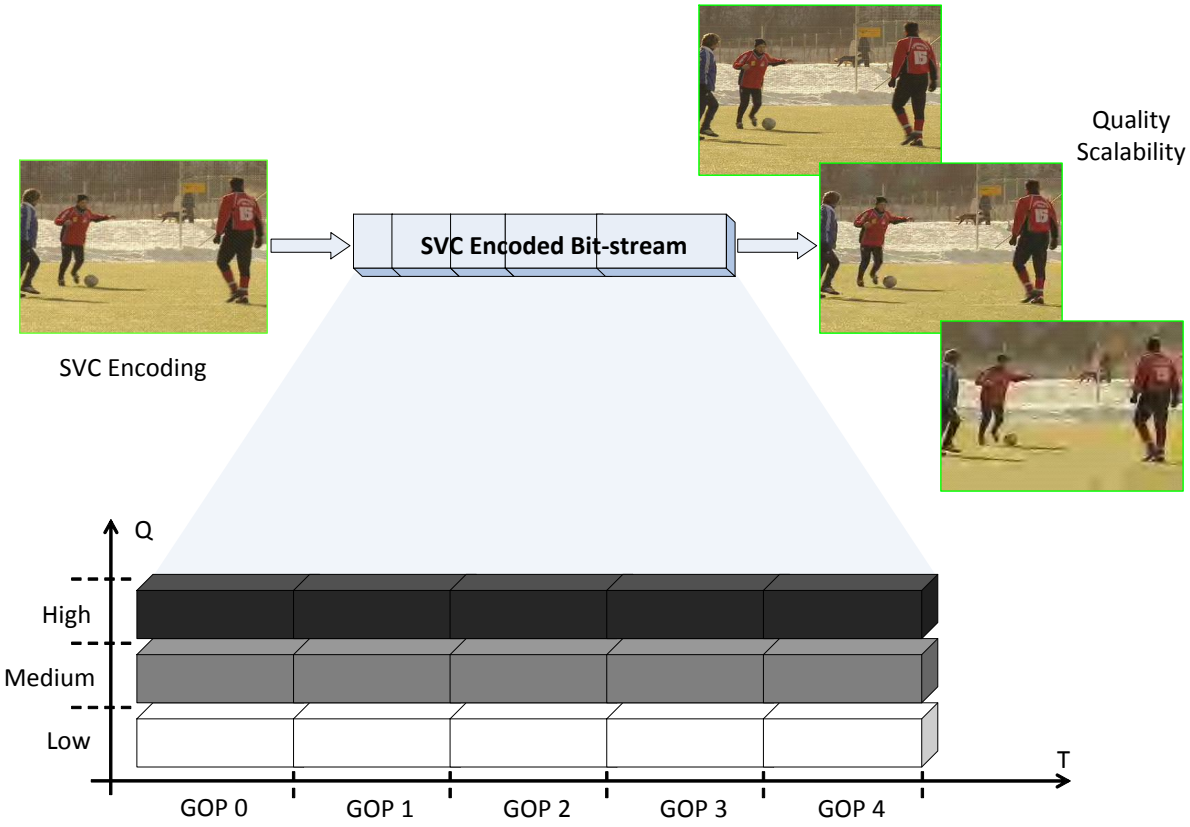


Figure 2.7: Structure of scalable video bit-stream when only quality scalability is considered; the video sequence is divided into GOPs and layers.

– corresponding to a Low quality in Figure 2.7 – needs to be completely received to decode a certain GOP. Therefore, it is the most critical part of the bit-stream. All the other layers are *enhancement layers* and they are used to improve the received video quality. If this representation of the sequence is used and layers have the same bit-rate, the only differences in the codecs as seen from the outside lie in compression efficiency and subjective quality perceived by the final user, which will be analysed in the next chapter. An important remark is that SVC technologies will be used throughout the rest of this thesis, from Chapter 3 to Chapter 6.

2.4 Introduction to Social Networks

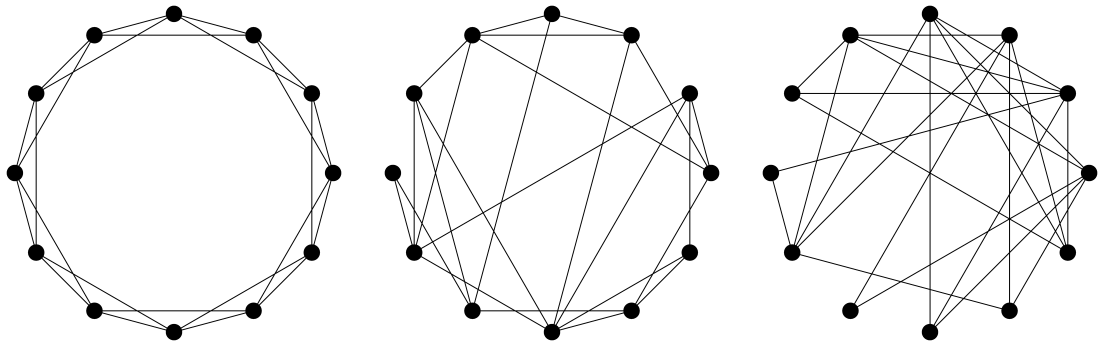
Social networking sites contribute nowadays to an important share of the Internet traffic. For example, the biggest social network, Facebook[1], as of early 2013 counts over 1 billion active members [39]. They can be defined as web-based services that allow users to [40]:

- *construct a public or semi-public profile within a bounded system,*
- *articulate a list of other users with whom they share a connection,*
- *view and traverse their list of connections and those made by others within the system.*

An important aspect emerging from this definition that can be applied to P2P is that, as we will see, creating a user profile contributes to the de-anonymisation of the network and this might be an incentive for users not to cheat [9]. Moreover, if users share a connection, they have the faculty to exchange information with the others, especially friends or in general peers they trust. An important remark is that friendship on a social network does not necessarily mean an actual connection in real life. Moreover, recent studies indicate that a significant percentage of profiles on Facebook (nearly 9% in 2012) are fake [41], therefore trust in social networks cannot always be implied. However, in principle, misbehaviour of a friend user is easily identifiable [9] and such users can be removed from the set of trusted peers.

Another characteristic that is peculiar of social networks is their topology, which can be defined as *small-world* [42]. That is, a few users have a number of connections that is significantly higher than the rest of the network and the maximum degree of separation between two random nodes in the network is small (if we assume all the nodes to be connected). In the real world, according to an old study [43], if two random persons are selected they will be connected (in the majority of the cases) by at most 5 intermediaries. This theory is more commonly known as *six degrees of separation*. It is however outdated, as a more recent study [44] performed on the network of active Facebook users indicates that the average number of nodes between two users is 3.74, suggesting at most four intermediaries, with small variations when national sub-graphs are considered.

[1] `https://www.facebook.com/`



(a) Regular, $p = 0$ (b) Small-world, $p = 0.5$ (c) Completely Random, $p = 1$

Figure 2.8: Effects of rewiring probability p on the topology of a Watts-Strogatz graph.

In 1998, Watts and Strogatz proposed a method to generate these type of graphs [42]. First of all, let N be the number of vertices and K the number of edges per vertex, the algorithm generates a ring lattice where each node is connected to $K/2$ neighbours on each side. Second, each edge is rewired at random with a probability p. Figure 2.8[2] shows that p influences the properties of the graph, from perfectly regular to completely random, whereas small world networks lie in between. A requirement for the parameters used in the graph generator is expressed in Eq. (2.1):

$$N \gg K \gg \ln(N) \gg 1, \tag{2.1}$$

where $K \gg \ln(N)$ is required to obtain a connected graph.

This model can be used to generate networks with a small world topology, however it shows some limitations. First of all, the degree sequence of the generated graph is not power-law; that is, the number of nodes with degree κ is not proportional to $\kappa^{-\beta}$ for any $\beta > 0$ [46], as expected for large κ from a real social network. Second, the model is not able to capture the property of densification [47] over time. This is a phenomenon that consists in an increase of the average degree of a graph as time passes. It also follows a power-law pattern; that is, given the number of nodes over time $n(t)$ and the number of edges over time $e(t)$, $e(t) \propto n(t)^{\alpha}$, with $1 < \alpha < 2$. Nevertheless, this model has been used in our approach in Chapter 6 due to its simplicity and as a starting point for future work, where more realistic models will be considered.

[2]These figures have been generated with Pajek [45] software.

Albert et al. proposed an algorithm to generate scale-free networks [46]. The starting point of their work is the consideration that most real-world networks represent systems that grow over time with the addition of new nodes. Moreover, the probability that a connection is rewired in the original Watts-Strogatz model does not take into account the fact that the probability of connecting to a node should also depend on the node's degree; in other words, it should show a preferential attachment to those nodes that are already popular and therefore have a high degree.

The algorithm works as follows: the starting point is a small set of nodes consisting of n_0 elements. At each step, a new node is added and e new connections are created (with $e < n_0$). These new connections, however, are not created randomly, but the probability Π that the new node connects to a node i depends on its degree κ_i and it satisfies the property expressed in Eq. (2.2).

$$\Pi(\kappa_i) = \frac{\kappa_i}{\sum_j \kappa_j}. \tag{2.2}$$

It is intuitive to see that after t steps, this model generates a network that consists of $n_0 + t$ nodes and $e \cdot t$ edges. Numerical simulations indicate that the probability that a node has κ connections follows a power-law with $\beta = 3$. Finally, as the total number of edges $e \propto n$, this model does not take densification into account.

On the other hand, Leskovec et al. presented an approach called *forest fire model* [47]. Let us suppose that there already exists a small graph consisting of n_0 users and a new node i wants to connect to it. First of all, it connects to an *ambassador node* j, creating a link to it. The second step consists in generating two random variables, x and y geometrically distributed and depending on *forward* and *backward burning probability*. Node i then selects x out-links and y in-links of j, connecting to the $x + y$ nodes on the other side of them. This step is repeated considering the in- and out-links of these new neighbours, however nodes cannot be visited more than once.

Graphs generated using this algorithm present a few interesting properties. First of all, it allows newcomers to connect to popular nodes quickly no matter which node they select as ambassador, according to the "rich gets richer" paradigm, which can also be found in [46]. Second, it facilitates the generation of "communities", as new users tend to replicate the social circle of their ambassadors and their connections. Finally, it satisfies

Topic	Where it can be found
Subjective Video Evaluation:	Chapter 3
Peer-to-Peer:	
P2P Transmission	Chapter 4 to Chapter 6
Credit-based P2P	Chapter 5 and Chapter 6
Scalable Video Coding:	
H.264/SVC	Chapter 3
WSVC	Chapter 3 to Chapter 6
Social Networks:	Chapter 6

Table 2.1: Summary of cross-references for background topics.

the aforementioned densification power law property and the diameter of the network shrinks as the number of users increase. That is, the maximum distance between two random nodes in the network decreases when more users join the network.

2.5 Summary

In this chapter we provided background information about QoE and subjective video quality, P2P networking including BitTorrent and credit-based systems, SVC and social networks. These are the elements our work has been built on, and a more detailed summary of where they can be found later on in this thesis is shown in Table 2.1. In the next chapters, more specific techniques that are relevant to our contributions will also be explained. Before that, Chapter 3 will illustrate our objective and subjective evaluations of WSVC and H.264/SVC over P2P.

Chapter 3

Subjective Video Quality Evaluation of SVC over P2P

The aim of this chapter is to evaluate the performance of two scalable codecs – WSVC and the standard H.264/SVC – under different static and dynamic network conditions in terms of subjective quality, and study how this relates to objective metrics[1]. The rest of the chapter is organised as follows: Section 3.1 explains the motivation behind this study and provides some related background. Section 3.2 describes the methodology used for our evaluation, in particular: the test sequences, environment, types of tests the subjects had to perform and the different P2P scenarios. Section 3.3 shows both objective (in terms of objective quality, assessed with metrics widely used in the scientific community) and subjective results of our experiments. Finally, Section 3.4 concludes the chapter, discussing the importance of these results in the design of P2P systems.

3.1 Motivation and Related Work

As the final users of digital content are actual persons and not machines, objective metrics used to determine video quality only give a partial assessment of how a video is actually perceived by the Human Visual System (HVS). Therefore, in the field of multimedia applications the necessity to carry out rigorous subjective tests for the assessment of video quality has arisen. These tests require a sufficient number of subjects and need

[1]The research work presented in this chapter has been developed within the FP7 project Saracen [48–52]. However, the subjective evaluation has been re-run.

to be carried out in a controlled environment, in order to guarantee the validity of the results and their reproducibility. In addition to this, test materials should be selected to cover several types of content and tests should be designed to reduce the impact of human bias to a minimum.

Objective metrics are used to express mathematically how much the original signal differs from the compressed one, in order to evaluate the performance of the encoding algorithm. Two very common tools used for objective video quality assessment are the Mean Squared Error (MSE) and the Peak Signal-to-Noise Ratio (PSNR), which will be described in detail in Section 3.3.1. A few studies [53, 54] indicate that there is not perfect correspondence between these metrics and how the HVS perceives video quality, especially if compression introduces artefacts. Several studies, which try to model the behaviour of the HVS [55, 56], have been proposed in the attempt to find a correlation between objective and subjective metrics. However, none of these models have proven to be fully reliable under a wide range of circumstances. Therefore, as both MSE and PSNR represent nowadays the most commonly used tools to evaluate the performance of video compression techniques, it is important to find in which cases these objective metrics can also be used to estimate how the HVS perceives a visual stimulus.

The study in [57] proposes a function that correlates objective and subjective metrics, expressed by Eq. (3.1):

$$QoE = e^{-c_i MSE}, \tag{3.1}$$

where the optimal value of c_i, found experimentally, is 8.05×10^{-3}. This function, however, does not always prove to be accurate, especially for different frame sizes and frame rates. In our study, we will conduct both types of tests and we will draw conclusions based on the significance of our experimental evaluation. However, it is out of the scope of this research work to model a function that correlates these two types of metrics.

As far as our subjective tests are concerned, international standards [58, 59] can provide guidelines to define such test activities. Tests can be classified according to the types of stimuli the subjects are presented with and fall into three categories:

- **double stimulus**: subjects are presented with a reference and a test stimulus sequentially. It can be required to rate the test only or both stimuli;

MOS	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

Table 3.1: Mean Opinion Score

- **single stimulus**: only a test stimulus is shown and the subject is asked to rate it;
- **stimulus comparison**: both stimuli are shown and the subject is required to assess relative quality.

Ratings can be expressed on a continuous or discrete scale, can be numerical or categorical (e.g. very bad to excellent) and represent the quality or impairment perceived by the subject when presented with the corresponding stimulus. The most commonly used metric used for subjective tests is Mean Opinion Score (MOS), defined by the standard [58, 59] and shown in Table 3.1 for the discrete case. Stimuli subjective evaluation can be performed while this is being presented to the subjects or at the end of the test sequence, in order to assess the impact of variations in the quality. Based on the degrees of freedom previously introduced, several types of assessments can be defined. For example, DSCQS represents a *double stimulus continuous quality scale* test, where an encoded test sequence and its reference are shown sequentially and the procedure is repeated twice. Ranking is performed after the second round and the difference between the MOS – expressed on a continuous scale from 0 to 100 – of the reference and compressed videos is calculated. Another method is DSCS, which means *double stimulus comparison scale*. It is similar to the previous test, the main differences being that both stimuli are shown only once and that a discrete scale is used.

Different types of tests are used for different purposes. For example, stimulus comparison immediately highlights the artefacts introduced by video compression, and it is used to evaluate the performance of the video codec regardless of its specific applications. On the other hand, double stimulus is more similar to a video transmission context, where the reference is not transmitted. Therefore, we used the latter approach

Ref.	Codec(s)	Sequences	Subjects	Methodology
[61]	MPEG-4	4	120	SC
[62]	H.263	5	20	DSIS
[63]	H.264/SVC	6	30	DSCQS
[64]	H.264/SVC	8	20	ACR
[65]	H.264/SVC, WSVC	3	16	SC

Table 3.2: Summary of related studies for scalable video codecs.

for our tests. However, we did not use a single stimulus as in the double stimulus case the difference between the perceived quality of the received and uncompressed video is more evident, making this test less vulnerable to the perceived quality of the original sequence.

The guidelines [58, 59] also provide tools to compensate for differences between subjects and calculate average MOSs and their corresponding confidence intervals. Other statistical tools can be used for a more thorough analysis of the data and comparison of results. An important remark is that subjective tests with scalable videos are usually performed to assess perceived quality and impairment associated to quality scalability. The main question is what is the minimum threshold for video quality to be considered as acceptable, and the answer depends on many parameters, such as content, type of application and viewers. The main studies regarding scalability for MPEG-4 [60], H.263 [37], H.264/SVC [5] and wavelet-based WSVC [32] are reported in Table 3.2, where SC is *stimulus comparison*, DSIS is *double stimulus impairment scale* and ACR is *absolute category rating*.

More specifically, [61] proposes an *optimal adaptation trajectory* to adapt video parameters to the current network conditions in order to maximise the perceived video quality. In fact, there are many possible choices to decrease a video bit-rate in case of a reduction in the current available resources: for example, video frame rate and frame size can be downgraded individually or at the same time and there are a large number of configurations that result in videos lying in the same zone of *equal average bit-rate*. Therefore, the authors propose a path in the *(frame rate, frame size)* space, which the adapting module should follow when choosing the video parameters, corresponding, as

said, to the minimum perceived impairment for a specific bit-rate. An important remark is that this trajectory also depends on the content of the video itself. Similarly, the study in [62] assesses video quality at low bit-rates, identifying equal MOS contours either in a *(frame rate, frame size)* or a *(SNR, frame rate)* space, concluding that the quality depends on the video content and that a small frame size is usually preferred under this low bit-rate constraint.

The authors of [63] propose a study on subjective video quality for mobile devices, analysing the impact of degradation in the spatial, temporal and SNR domain, showing how different types of sequences are affected. For example, the authors found that a cartoon sequence is not significantly affected by a frame rate reduction. They also identified a few scaling order preferences associated to different type of content. For instance, for a sequence showing a Sports event, the subjects preferred a reduction in the frame rate to a SNR degradation. The authors justified this phenomenon in terms of user expectation rather than the characteristics of the video themselves.

In [64], Song et al. analyse the relationship between video content and perceived quality, suggesting that different segments of long video sequences should be encoded with different parameters to maximise perceived quality. They also recommend that studies should not just focus on the intensity of motion (e.g. high or low), but also on the size of the moving area with respect to the whole picture. Moreover, given a fixed spatial layer, the results of this work suggest that SNR scalability should be prioritised with respect to temporal. On the other hand, for extremely low frame rates, spatial scalability should have priority over temporal. Finally, the evaluation performed in [65] compares H.264 and WSVC, showing that the preferences between a degradation in the spatial resolution and the frame size depended on the content of the test sequences and that the behaviour of the two codecs is overall consistent, despite the fact that the type of distortion introduced by the two encoding techniques is quite different.

In our study, we will focus on SNR scalability, as our proposed P2P systems will mostly scale video sequences with respect to this parameter. Finally, despite the study in [65] also focussing on WSVC, it is based on stimulus comparison and does not consider scenarios with variable quality.

3.2 Subjective Evaluation Methodology

In this section we clarify the methodology we used for our subjective evaluation of scalable video over P2P. First, we give a short description of the test video sequences. Second, we describe the characteristics of our controlled environment. Finally, we describe the scenarios we defined to simulate different behaviours of the P2P network when transmitting scalable video sequences.

3.2.1 Test Sequences

For our subjective video quality evaluation, we used five uncompressed, high quality, test sequences showing different types of content. All of them are real scenes and there are no computer generated scenes nor animations. All used test sequences are professionally acquired and show no distortion. The original content is High Definition (HD), in YUV 4:2:0 format, not interlaced and it does not include any audio. The sequences have been downsampled to 4CIF (704x576 pixels) resolution while keeping the same aspect ratio, to avoid introducing distortion. A 10-second interval from each sequence has been encoded in both H.264/SVC and WSVC format using 4 layers and the encoded videos are the same as the ones used in Saracen project [48]. Their layers roughly have the same bit-rates; despite their differences being negligible, these values are not exactly the same, as they are the result of the choice of other quantisation parameters. The bit-rate of the layers depends on the sequence itself, as different activity results in a different quality when the bit-rate is fixed and the bit-rates had to be chosen in such a way that there was significant visual difference between the layers. All sequences have a frame rate of 30 fps and the GOP size is 16, hence there are 16 GOPs in each video. The content of the sequences can be summarised as follows, while a frame for each sequence is shown in Table 3.3:

- **InToTree**: Camera pan, shows a building and a tree.
- **PersonsReporting**: Still camera, shows two persons reporting news as if they were anchormen.

Slow Motion	
	
a) InToTree	b) PersonsReporting
Medium-intensity Motion	
	
c) PrimeTelFootball	
Fast Motion	
	
d) Soccer	e) JohnnyTrailer

Table 3.3: Sample frames from the test sequences used for subjective evaluation.

Sequence Name	Spatial Activity (SA)	Temporal Activity (TA)
InToTree	8.59	12.64
JohnnyTrailer	10.00	81.49
PersonsReporting	12.00	4.00
PrimeTelFootball	17.41	47.28
Soccer	10.20	51.28

Table 3.4: SA and TA of test sequences used for subjective evaluation.

- **PrimeTelFootball**: Camera pan, shows the public inside a football stadium and the field.
- **Soccer**: Camera pan, shows a few people playing football. This sequence is a standard video testing sequence and has also been used for other experiments in this thesis.
- **JohnnyTrailer**: A ten-second extract from "Johnny English" movie trailer, with fast action, sudden changes of the scene and text.

These sequences can be characterised by their spatial activity (SA) and temporal activity (TA), which depend on the amount of details and motion they contain respectively. These values have been computed as indicated in Eq. (3.2) and Eq. (3.3) [59]:

$$SA(n) = rms_{space}[Sobel(I(n))], \tag{3.2}$$

$$TA(n) = rms_{space}[I(n) - I(n-1)], \tag{3.3}$$

where rms_{space} indicates the root mean square computed over the entire frame, $I(n)$ indicates the frame at time n and $Sobel()$ indicates Sobel filter [66], which is used for edge detection and reveals the presence and location of high-frequency components in the frame. An average for TA and SA has been computed over the selected ten seconds for each sequence and is reported in Table 3.4. The table shows that for these tests we have used two sequences with high TA (Soccer and JohnnyTrailer), two with low TA (InToTree and PersonsReporting) and one with high SA and low TA (PrimeTelFootball).

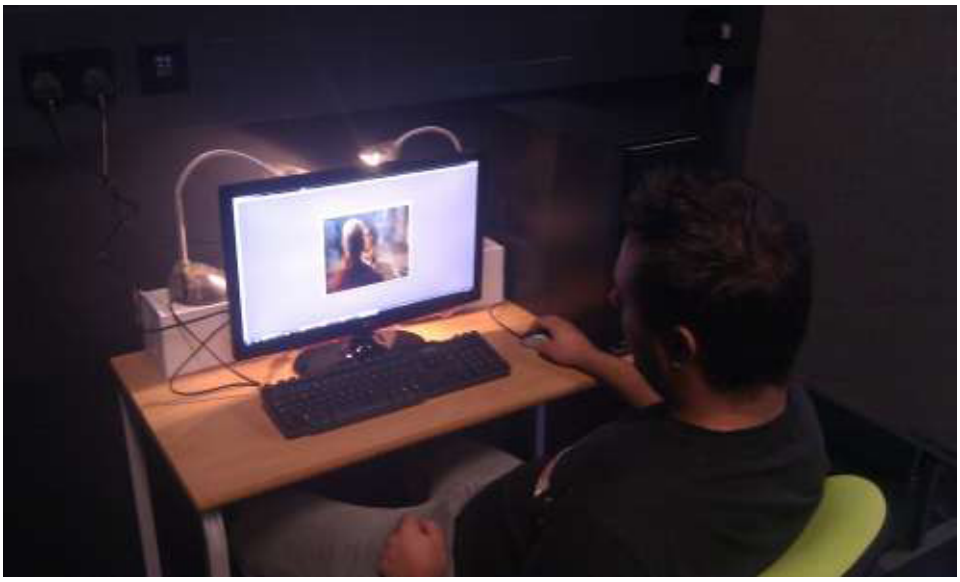


Figure 3.1: Illustration of the testing environment.

3.2.2 Testing Environment

The aim of using a controlled environment, such as a test room, is to assure the reproducibility of the subjective test activity by avoiding the involuntary influence of any external factors and following the instructions in the guidelines [58,59] serves this purpose. These tests were performed at the Multimedia and Vision Group (MMV) laboratory at Queen Mary University of London and the testing environment was set up according to the recommendations specified in [58,59]. The test room was equipped with two Desktop PCs, each one with a LCD monitor. The ambient lighting consisted of neon lamps with 6500 K colour temperature and halogen lamps. The wall colour was black. One subject at a time sat in front of one of the LCD screens at a distance of about two to three times the diagonal size of the stimulus, and it was chosen specifically for this application. It is within the range of 1 to 8 times the height of the picture height, as expressed by the standard [59]. A picture of the test room is shown in Figure 3.1.

3.2.3 Subjective Testing

The method consisted of pair-wise comparisons between two stimuli (test sequences). The subject was asked to indicate the perceived quality for each, on a continuous scale

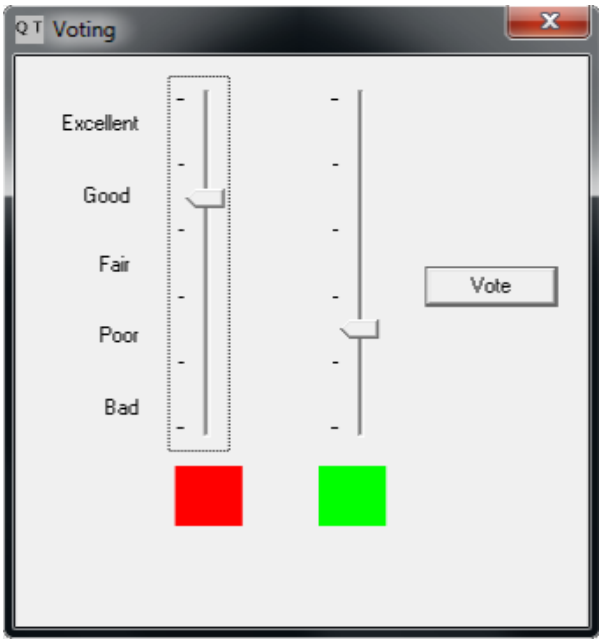


Figure 3.2: Subjective Quality Continuous Scale

from "Bad" to "Excellent", as shown in Figure 3.2 and the ranking was expressed on a scale from 0 to 100. The subjective test proceeded as follows. Fifteen pairs of test sequences for each scenario were played one after another. The videos in the pair were chosen randomly, however, one of the sequences was always the original uncompressed bit-stream, while the other could be a H.264/SVC video, a WSVC video or another display of the original sequence.

To limit the duration of a test session, the stimuli presentation was divided into two separate sessions, each of which consisted of three scenarios. Prior to the test sessions, a training session took place, where the test methodology was described to the subject by using a set of training stimuli different from the test stimuli. Eighteen subjects (11 men and 7 women) participated in the experiment, above the suggested value of 15 [59]. They were recruited among a group of postgraduate students and three of them were experts in video coding. They reported normal or corrected to normal vision. The average age of the subjects was 26.

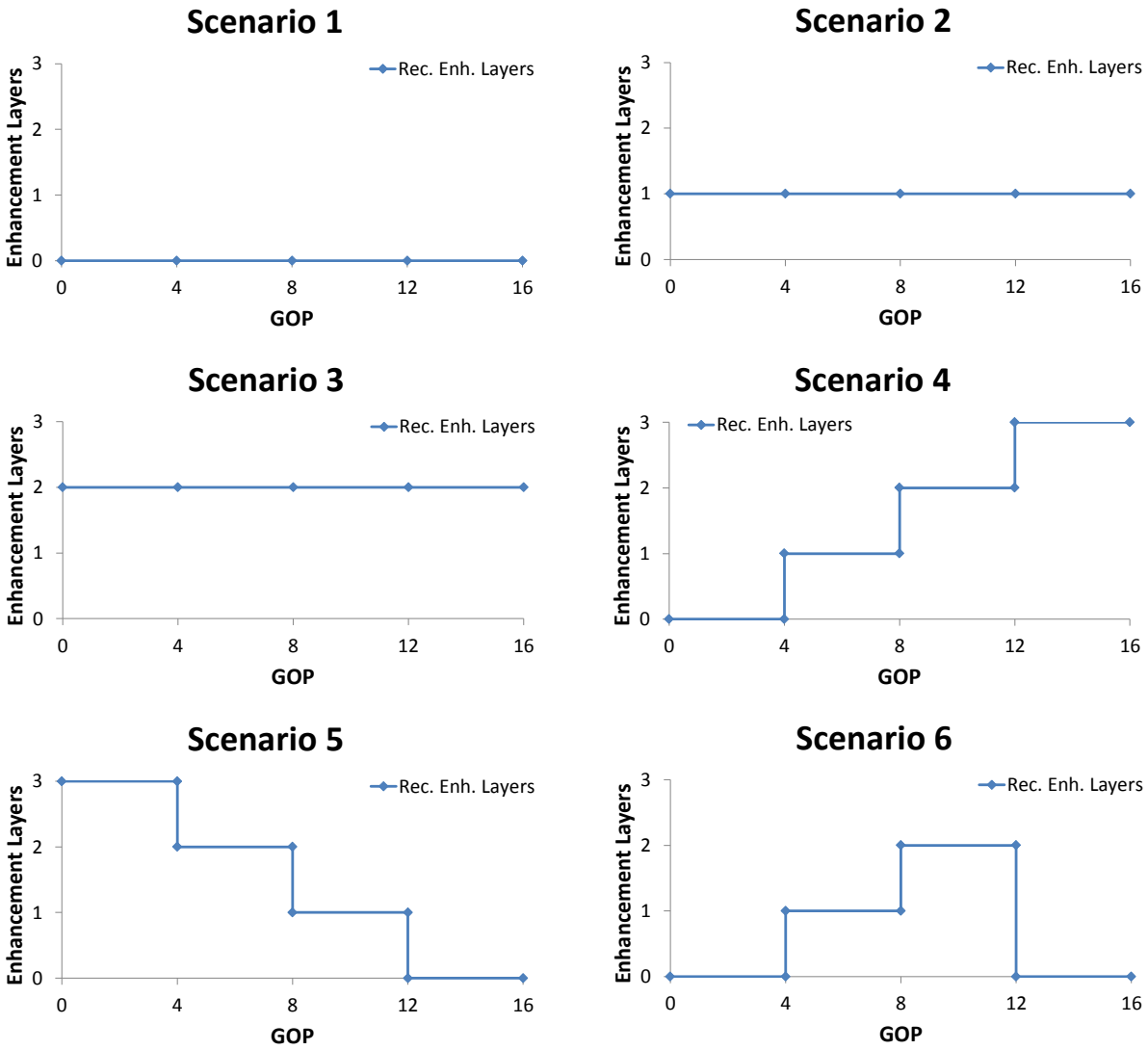


Figure 3.3: SVC over P2P scenarios for subjective video evaluation.

3.2.4 Scenarios for Subjective Evaluation of P2P Transmission

For subjective evaluation, six different scenarios were considered. They were designed to simulate actual potential behaviours of the P2P network and they can be illustrated as follows:

Scenario 1 Only the base layer of each GOP has been decoded.

Scenario 2 The base layer and one enhancement layer have been considered.

Scenario 3 The decoded sequence consists of the base layer and two enhancement layers.

Scenario 4 Only the base layer is considered for the first four GOPs and the number of additional layers increases to three.

Scenario 5 Symmetrically, the first four GOPs consist of base and three enhancement layers, after which the number of enhancement layers received gradually goes down to zero.

Scenario 6 The first twelve GOPs of each sequence are the same as Scenario 4, while the last four only contain the base layer.

Moreover, Figure 3.3 shows the different scenarios with respect to the number of quality layers for each GOP on a scale from 0 to 3, where "0" indicates that only the base layer of that GOP has been received and "3" denotes three enhancement layers.

Scenarios 1 to 3 might prove valid over a period of time when the network is relatively stable. In P2P networks at each time the sum of all the download rates of all the nodes in the network needs to be equal to the sum of all their upload rates, as it will be explained in Section 5.4.2. Therefore, regardless of any consideration about fairness and cooperation in such networks, different download rates may be experienced depending on different capabilities of each user. If the network resources are scarce, users might be able to download only the base layer of the test sequence, like in Scenario 1. On the contrary, if nodes with higher capability are present, or if additional servers are used as data seeders, users might experience higher download rates, leading to Scenario 2 or 3. Another possible case is represented by different costs associated to download. For example, if a peer wants to join the P2P network using a mobile device over a 3G connection, it might be only interested in the base layer of the sequence, as it might have to pay in relation to the amount of downloaded data. The same user might want to download a higher resolution version (base and one or two enhancement layers) when connected over WiFi. On the other hand, Scenarios 4 to 6 correspond to nodes with higher capability joining and recovering (Scenario 4) or leaving (Scenario 5) the network, or a combination of these two cases (Scenario 6). In addition, Scenario 5 might also happen as a consequence of node failures, or as a consequence of sub-optimal network resources allocation, e.g. if a user is not using all of its uploading bandwidth, but has no additional data it can contribute to its neighbours.

These scenarios do not take pauses in the video playback into account and only consider 10-second sequences. Assessing the impact of playback interruptions – especially in comparison with very low bit-rate video – on subjective quality and performing tests on longer sequences will be addressed as future work.

3.3 Results

In this section we present the results from both our objective and subjective evaluations. First, we explain the metrics used for the objective evaluation and show the corresponding obtained graphs. Second, we present and analyse the subjective results, obtained in the aforementioned P2P scenarios.

3.3.1 Objective Results

The metrics we used for our objective evaluation are MSE and PSNR, already introduced in Section 3.1. These tools are widely used by the video processing community to evaluate the performance of lossy video codecs. One of the reason for their success is their simplicity and ease of implementation, where at the same time they are objective metrics used to quantify the distortion introduced in the encoding process. PSNR is a function of the ratio between the maximum value a signal can have and the "noise" associated to compression, which is represented in this case by the MSE. It is usually expressed in decibels and is calculated on the luminance component of the video. Therefore, a higher PSNR is better, as it implies a lower error. For a single frame, MSE and PSNR are calculated as shown in Eq. (3.4) and Eq. (3.5):

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left[f(i,j) - \tilde{f}(i,j)\right]^2}{M \cdot N}, \tag{3.4}$$

$$PSNR = 20 \cdot \log_{10}\left(\frac{255}{\sqrt{MSE}}\right), \tag{3.5}$$

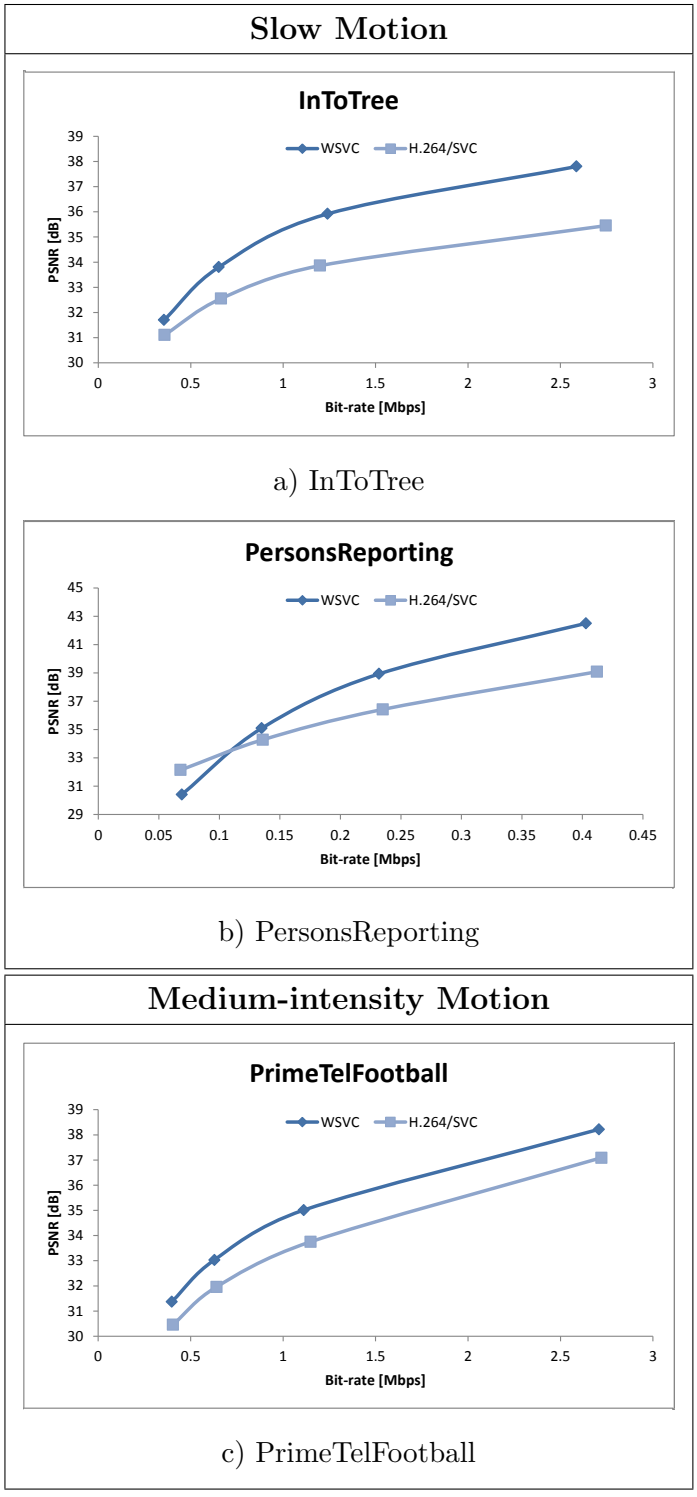


Table 3.5: PSNR for slow and medium-intensity motion sequences; 4-layer WSVC and H.264/SVC.

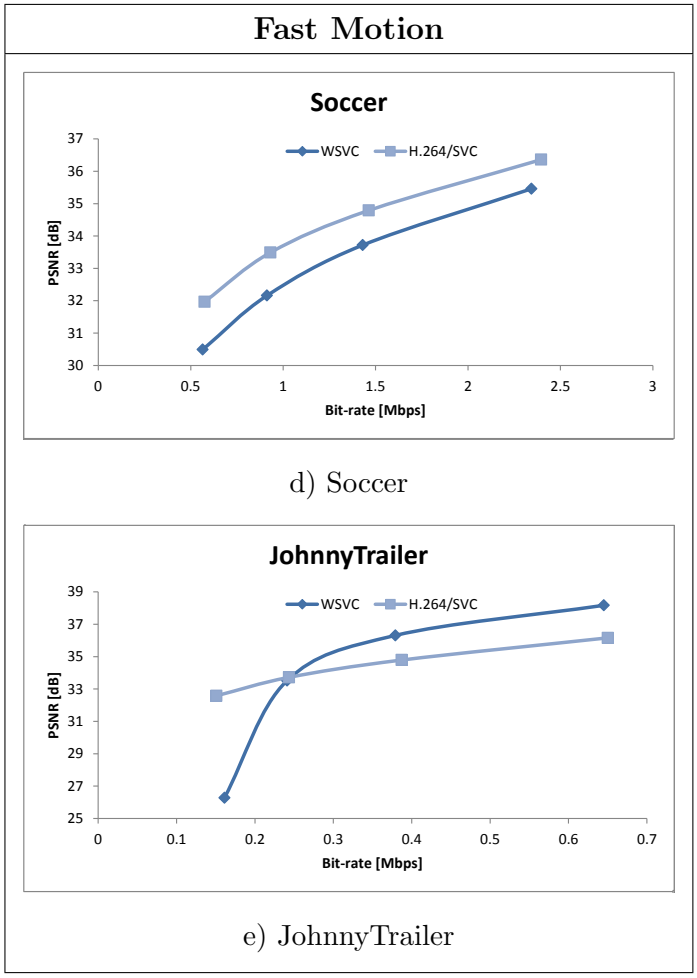


Table 3.6: PSNR for fast motion sequences; 4-layer WSVC and H.264/SVC.

where $M \cdot N$ represents the total number of pixels in the frame, $f(i,j)$ represents a pixel from the original image, $\tilde{f}(i,j)$ is the reconstructed image and 255 is the maximum (peak) luminance value, provided that it is represented using 8 bits. The sequences have been encoded with the aforementioned codecs, H.264/SVC and WSVC, using bit-rates that were as close as possible to guarantee a fair comparison, and have been adapted to all four possible extraction points, e.g. removing part of the original encoded bit-stream. Table 3.5 and 3.6 show the PSNR of the test sequences at these extraction points. An important remark is that sequences with low TA and SA achieve good performances at low bit-rates, as there is strong correlation between frames and less energy associated to high frequency coefficients in the spatial domain, whereas sequences with fast motion or high detail require a much higher bit-rate to achieve the same performance.

The results show that for slow moving sequences (with low TA) WSVC performs better than H.264/SVC, whereas for high moving sequences (high TA) the roles are inverted, especially at low bit-rates. Moreover, at higher bit-rates, with the exception of Soccer (in Table 3.6), WSVC shows either comparable or better performance with respect to H.264/SVC for all types of sequences (low, medium and high TA).

3.3.2 Subjective Results

Table 3.7 and 3.8 show the average MOS for Scenarios 1 to 6 for the five test sequences, which have been grouped according to their motion content.

First of all, we will consider Scenario 1 to 3, where bit-rates do not change throughout the playback. As expected, the average perceived quality increases as the bit-rate and PSNR increase. This difference is statistically significant in most cases, and the biggest difference can be found in fast moving sequences. For the cases in which this difference is not statistically relevant, it is plausible that the difference in the encoding bit-rate is too little to have a significant impact on the video quality perceived by the subjects.

As far as slow moving sequences are concerned (PersonsReporting and InToTree), Table 3.7 indicates that WSVC performs as good as H.264/SVC when only the base layer has been decoded and better than the other codec when two enhancement layers have been received. In the case when only one enhancement layer has been received, InToTree has a higher MOS when encoded in WSVC format, while the performance is similar for PersonsReporting. On the other hand, for fast moving sequences (JohnnyTrailer and Soccer), H.264/SVC outperforms WSVC at all bit-rates, as shown in Table 3.8. The graphs suggest that for H.264/SVC receiving two enhancement layers, given the current bit-rate parameters, does not have a much bigger impact than receiving only one, as it does not result in a much bigger gain as compared to only receiving the base layer. For WSVC, on the other hand, the increase in the perceived video quality is much more constant. Finally, for the medium-moving sequence (PrimeTelFootball), the graph in Table 3.7 indicates that the perceived quality does not show a significant difference depending on the codec used – even if the average MOS is bigger for WSVC at the highest bit-rate – and therefore we can say that for this video the performance of WSVC

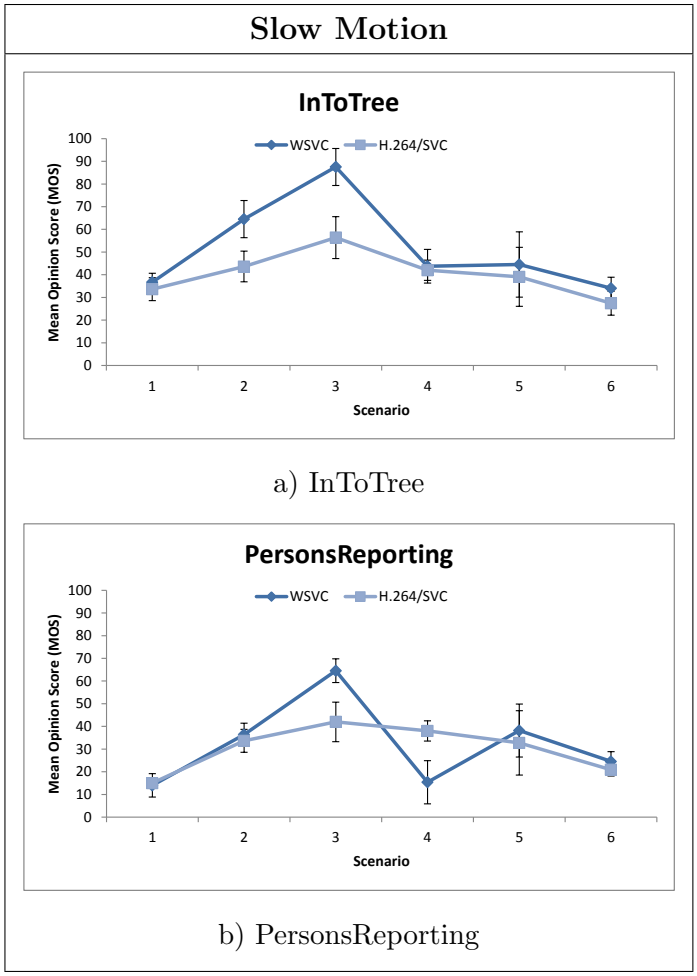


Table 3.7: Mean Opinion Score for slow and medium-intensity motion sequences, all scenarios, WSVC and H.264/SVC.

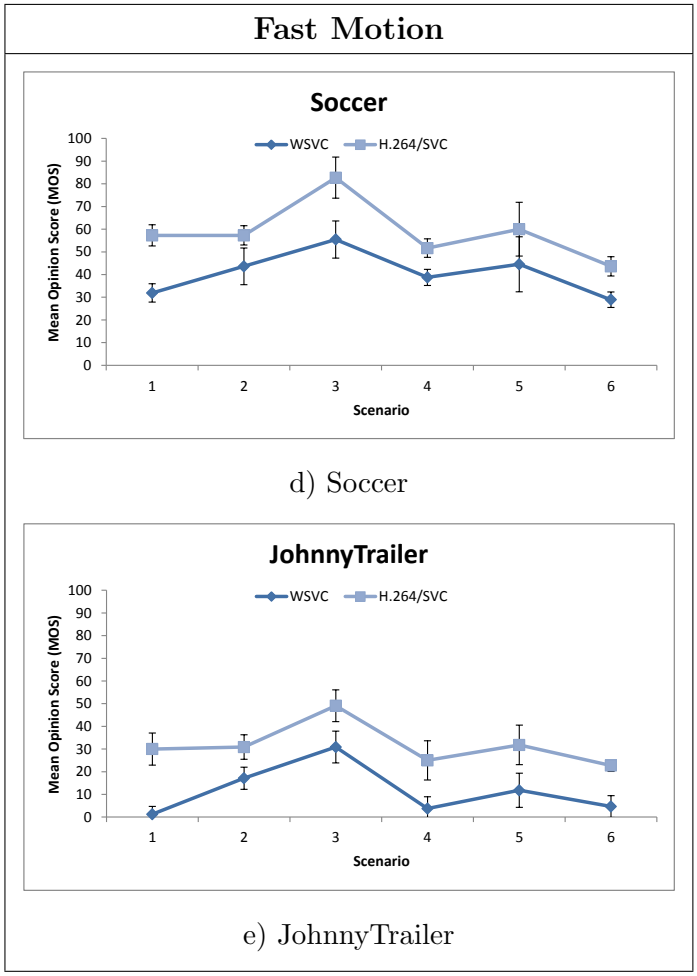
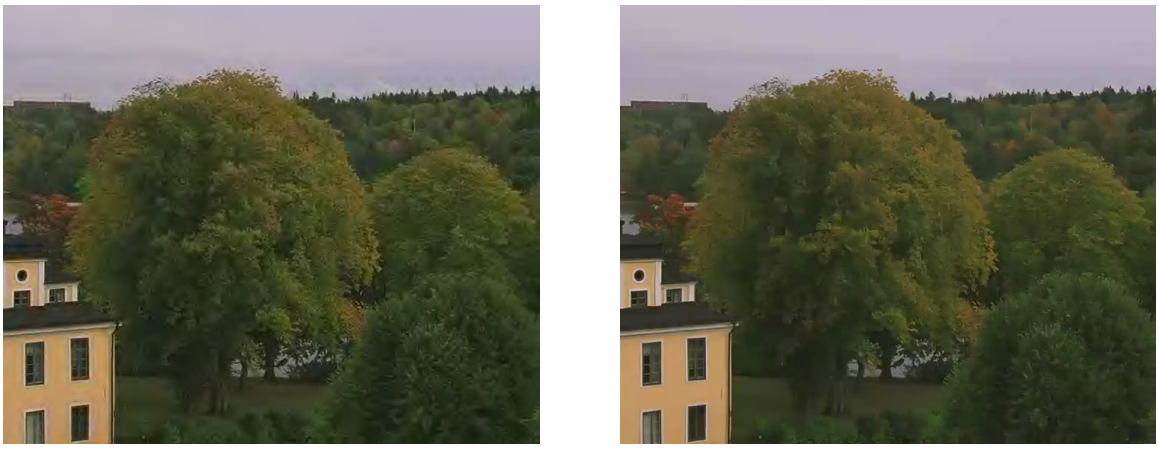


Table 3.8: Mean Opinion Score for fast moving sequences, all scenarios, WSVC and H.264/SVC.

and H.264/SVC is similar. An important remark is that these results are coherent, as expected, with our objective evaluation, with the exception of JohnnyTrailer. Our explanation for this behaviour is the different types of artefacts introduced by the two codecs; for example, the users might be more accustomed to the type of compression introduced by the standard codec. Two frames showing the impact of the two different encoding techniques are shown in Figure 3.4.

Scenarios 4 to 6 correspond to more dynamic situations that may occur during scalable video transmission over P2P. The average MOS for these scenarios are also shown in Table 3.7 and Table 3.8. Overall, the scores obtained by the two codecs in Scenarios 4 and 6 are much lower than those found in Scenario 2, despite the average bit-rate being



(a) WSVC (b) H.264/SVC

Figure 3.4: Sample frames from InToTree; only the base layer has been decoded.

higher, while Scenario 5 is roughly comparable with Scenario 2 (despite the uncertainty on the results being bigger). Therefore, these results suggest that the subjects did not like changes in the video quality, however were positively impressed by a high video quality at the beginning of the playback, which could be achieved for example by extending the pre-buffering time. This result may however be a limitation of the sequences used for testing, as they are 10-seconds long. Previous literature studies show that sudden quality degradation can significantly affect the perceived video quality [61], and a recent solution proposed in the context of Dynamic Adaptive Streaming over HTTP (DASH) [67] aims at using intermediate quality layers to smoothen the transition from high to low quality, which results in an overall improvement of the performance. Scenario 5 in our study also presents a smooth transition and in our opinion this is the reason why the MOS is higher than expected.

An important remark is that the behaviour observed in these scenarios is the same for both codecs, which suggests that both WSVC and H.264/SVC are affected by PSNR variations in roughly the same way. Finally, even for the variable scenarios the results suggest that H.264/SVC performs better than WSVC for fast moving sequences, while for slow and medium moving sequences the performances of the two codecs are comparable in most cases.

3.4 Conclusion

In this chapter, we analysed scalable video transmission over P2P from a subjective video quality perspective. Before conducting the subjective experiments, we performed an objective analysis using the most advanced PSNR metric. According to this evaluation, WSVC shows better results for slow moving sequences, while H.264/SVC performs better for fast moving sequences, in particular at low bit-rates. On the other hand, at higher bit-rates WSVC shows equal or better results with respect to H.264/SVC for both slow and fast moving sequences, the only exception being Soccer.

As far as our subjective evaluation is concerned, we found that in general, for slow moving sequences, there was either no significant difference (for low bit-rates) or WSVC performed better (at higher bit-rates) as compared to H.264/SVC. This is coherent with what we already found in our objective evaluation. For fast moving sequences, H.264/SVC performed better as compared to WSVC, especially at low bit-rates. For medium moving sequences, there was no significant difference between the two codecs.

Regarding the scenarios with constant received bit-rates, we observed that in general, as expected, the performance of both codecs improved as users had successfully downloaded more layers at the receiving end, through the P2P network. Hence, the protocol designer needs to make sure that all the users of the system receive the highest possible layer according to the given network conditions. On the other hand, this situation radically changes when the number of decoded layers is not constant. In fact, Scenario 2 received higher perceived quality score as compared to Scenarios 4 and 6, while the scores assigned to the videos in this Scenario were comparable with Scenario 5. This result confirms that frequent quality changes are not seen favourably by the users, and lower yet constant received video bit-rate will result in a better user experience. However, quality will be overall higher if the user receives the better quality at the beginning of sequence.

3.4.1 A Lesson Learned

Finally, as far as the findings in this chapter that will be used in the remaining parts of this thesis are concerned, they can be summarised as follows:

- The results obtained with WSVC codec have the same validity as those that would be obtained with H.264/SVC.
- It might be useful to increase the pre-buffering time to provide users with a better video quality since the beginning instead of changing the video quality, as users seem to be biased by a first-impression factor.
- The aim of chunk request policies for both H.264/SVC and WSVC should be to provide final users with a received video quality that is as constant as possible.

In the next chapter, we will describe a general block diagram which is valid for all our proposed architectures, then we will illustrate our proposed algorithms for piece picking and neighbour selection for P2P SVC transmission, which will consider the results found in this chapter.

Chapter 4

Scalable Video Adaptation to P2P Transmission

This chapter starts with the presentation of a block diagram showing a general architecture for SVC streaming over a P2P network. Subsequently, our first proposed approach is described. In summary, it consists of a piece picking and a neighbour selection policy, whose aim is to allow efficient scalable video transmission over a P2P network. In such a system, users experience different video qualities depending on their available resources. These techniques have been implemented as modifications of existing algorithms for non-scalable video. This is a basic solution, which handles scalable video streaming over P2P networks in such a way that the system is still compatible with the original Tribler [68] BitTorrent client, presented in Section 4.3.3. Therefore, resource allocation to other peers is performed according to the algorithm used by the original client, which will also be illustrated in Section 4.3.3. This chapter is organised as follows: the general social P2P architecture for SVC is presented in Section 4.1; the problem is introduced in Section 4.2; the related background is provided in Section 4.3; our algorithms are described in Section 4.4; Section 4.5 gives a short description of how the framework for testing these techniques has been implemented; Section 4.6 consists of the experimental evaluation; finally, Section 4.7 concludes this chapter, analysing the pros and cons of this solution.

4.1 Overview of a General Social P2P SVC System

Before explaining our proposed architectures, we describe a general block scheme for a system that performs transmission of scalable videos over a P2P network that presents social features. It will be used as a reference for the techniques illustrated in this chapter and in the following ones. First, we illustrate a diagram that shows the main components, and then we provide a short description for each of them.

4.1.1 Block Diagram

The scheme presented in this section is valid for each of the approaches described in this thesis, however, depending on the approach itself, some blocks might not be used, or might correspond to state-of-the-art technologies instead of new contributions. For the systems presented in this chapter, Chapter 5 and Chapter 6 we draw similar diagrams, highlighting where the novelties lie. The architecture is shown in Figure 4.1. In this and all the related diagrams, full lines indicate the exchange of video chunks, while the dashed ones denote control or information messages.

As P2P networking relies on the fact that users both receive and transmit data, we can identify the two most important blocks: one which is responsible for resource allocation (uploading) and one for chunk selection (downloading). This is not valid for all P2P systems (only for the *pull-based* ones), however, all our proposed techniques rely on this scheme. In general, downloading mechanisms need to be aware of the characteristics of the video sequence they are requesting. In other words, they need to know the structure of the codec. On the other hand, resource allocation can depend on a credit-based framework, and information received from reliable peers, such as friends in a social network. More specifically, these blocks can have the following functions:

- **Chunk Selection**. In this thesis, we will usually refer to this block as "piece picking policy". It is responsible for requesting chunks of data to other users. As our systems deal with video transmission, it needs to request these chunks with the aim of ensuring a satisfactory video experience for the final user. For example, it needs to make sure that video chunks are correctly received and decoded before their playback time, with the best quality a user can afford to download. Our

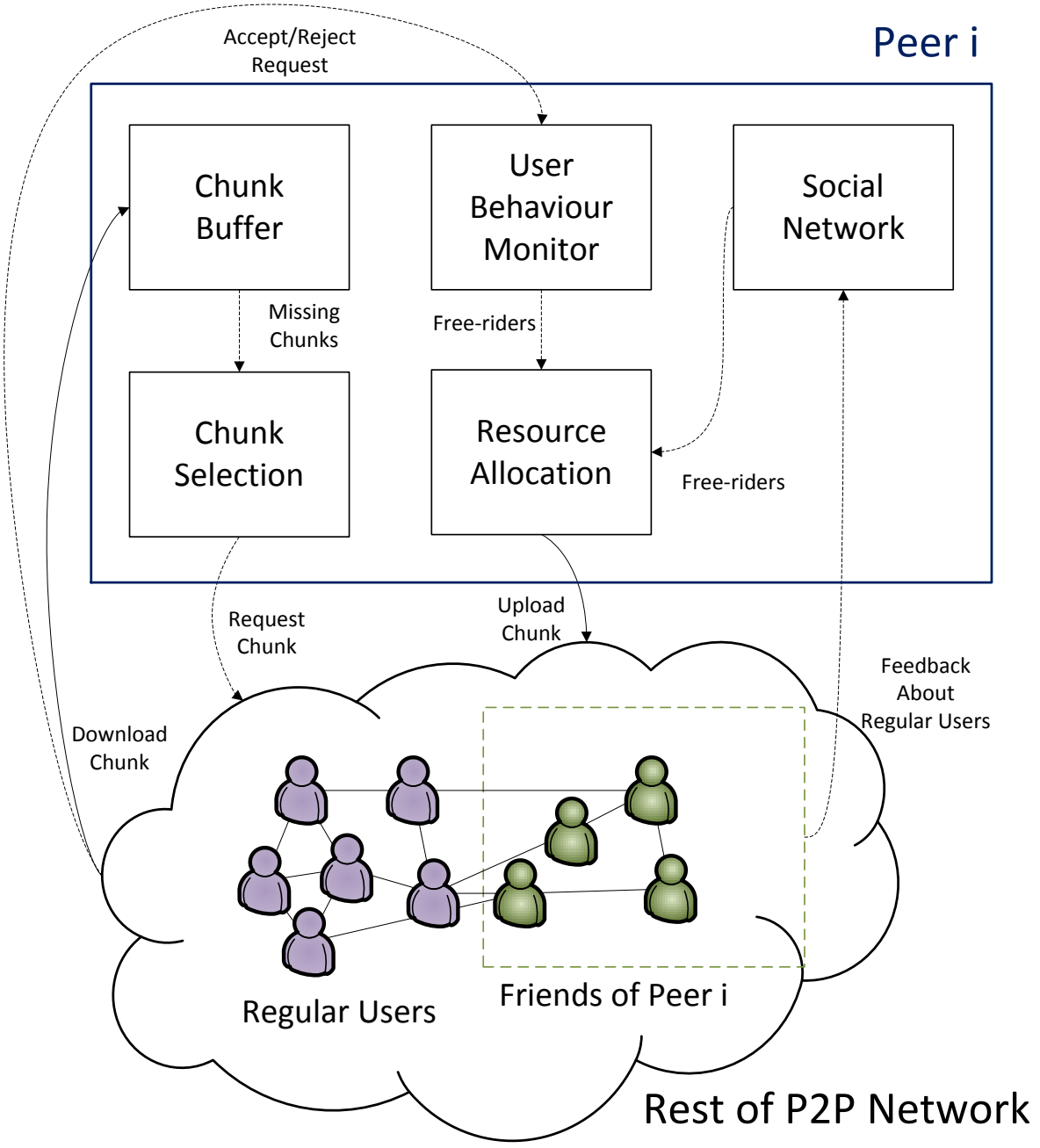


Figure 4.1: Generic block diagram for our proposed architectures.

piece picking policies work with scalable video. This allows to add flexibility to the system. In fact, some parts of a scalable video bit-stream can be removed and the new sequence still be decoded, albeit a downgraded version. This system needs information from:

  - **Chunk Buffer**. This module keeps track of the missing video parts. The Chunk Selection module will then generate requests depending on the information provided by this module (and the respective timestamps of the chunks).

- **Resource Allocation**. This is the most critical part in every P2P system. It is responsible for deciding which users a peer should cooperate with and the amount of this contribution. Optimising resource allocation (with the constraint of the upload capacities of the single peers) also means making more data available to the network. This can result in a "virtuous circle", as this data can be shared with even more users. In our proposed resource allocation modules, this decision always depends on a peer's current credit or information users can retrieve from their circle of trusted users. Credit-based P2P paradigms are often introduced to promote fairness among peers. We remark that if a system does not make use of rewards or punishments for good or deceptive behaviour, a peer has no interest in sharing their resources. This was a typical situation in the first P2P systems, whereas in the most recent ones, resource allocation mechanisms tend to reward users that have proven to be trustworthy. This module requires the information provided by:

  - **User Behaviour Monitor**. This module is responsible for collecting information about the interactions a peer has had with the rest of the network, and more generically about the network itself. This module is also responsible for the generation of some metrics, which will be evaluated by the Resource Allocation module when taking decisions.

  - **Social Networking**. The increase of popularity of social networks has contributed to the de-anonymisation of the Internet, as users are allowed to connect with other people they might know in real life. When a relationship among users that trust each other, such as "friendship", is established, they can share some information which is *de facto* verified. Within the context of P2P networks, each user tries to build a map of their neighbours, trying to understand whether they are cooperative or malicious. When friend users trust each other, they can combine their efforts and have a better understanding of the bigger picture. It is important, however, that peers choose their friends carefully, otherwise the system might be vulnerable to social attacks, where malicious peers ask for a someone else's trust, sometimes even pretending to be multiple users.

4.2 Problem Description

As far as our first approach is concerned, we propose a simple solution to allow transmission of scalable sequences over a P2P network. The starting point is the existing BitTorrent protocol, which is however not suitable for video streaming without proper modifications, as it will be discussed later in Section 4.3. The scalable sequences we use are encoded in WSVC [32] format, but this solution can be easily extended to H.264/SVC [5]. Despite this codec supporting spatial, temporal, SNR and combined scalability, as explained in Chapter 2, we will only consider quality scalability in this case.

More specifically, the issues we address in this chapter are the following:

- Define a *piece picking* policy whose aim is to download the video chunks that are following the current playback position, trying to maximise the received video quality given the current available resources.
- Propose a *neighbour selection* policy which identifies unreliable peers in the network, in order to avoid downloading base layer chunks from them.

4.3 Related Work

Given the description of the original BitTorrent protocol in Section 2.2.2, it is clear that it is not possible to transmit video sequences efficiently without introducing some modifications. For example, rarest-first piece picking policy is incompatible with sequential frame visualisation; in fact, a user has to wait for the download to be complete before being able to play the video. Moreover, T4T mechanism alone can result in poor transfer rates at the beginning of the download, which means that pre-buffering phase could be unnecessarily long.

4.3.1 Popular P2P Systems for Video Transmission

First of all, we describe a few very popular P2P systems that can perform either VoD or live streaming. None of them were originally designed for SVC transmission, however all these protocols can in principle be extended to support it.

Similarly to our proposed approach, PPLive [69] is an example of pull-based architecture for VoD, where users need to implement a piece selection and neighbour selection policy. As far as the chunk selection strategy is concerned, it is *mixed*, in the sense that sequential download has the highest priority, but rarest-first is also applied. Pieces are requested from a set of neighbours whose size is kept under control and peers have no control over their level of contribution, which implies that cooperation is enforced. In addition to SVC support, this is the main difference with our approaches described both in this and in the following chapters.

BulletMedia [70] is another solution where peers need to explicitly request data to other users. Video blocks are selected according to a *rarest random* strategy, however the base principle of this protocol is that there should be a minimum number of content replicas available at all times. Therefore, a peer not only requests blocks that are close to its playback position, but also other parts of the stream that are not currently being cached by a sufficient number of peers in the network. This is known as *proactive caching* and is only performed if the current playback can be sustained. Proactive caching could in principle be applied to our system as well, provided that there are enough spare resources available, however in our case peers will in general try to improve their received video quality, rather than downloading other parts of the stream.

On the other hand, RedCarpet [71] divides a video sequence in *segments*, which are subsequently divided into *blocks*. As far as chunk requests are concerned, peers use different policies when requesting data from a central server, which owns the first copy, and the other clients. As far as server policies are concerned, in addition to requesting a random piece in the segment, options include requesting: the overall rarest piece in the system, the system-wide rarest piece in the segment, the rarest piece in a peer's neighbourhood and the rarest piece in a peer's neighbourhood belonging to the peer's current segment. Client policies, on the other side, allow to request a random piece within a segment or the rarest piece in a peer's neighbourhood within the current segment. An important remark is that all the peers follow the same policy. The authors

propose many other policies, which also use network coding, however their main finding is that the system performs better if peers receive the globally rarest blocks, and not just the ones within the segment. As far as a comparison with our systems is concerned, we considered using a sliding window as opposed to dividing video chunks into groups a more appropriate choice, as the window keeps following the current playback position. Moreover, the experimental evaluation of this system shows huge variations in its performance over time, with minima when a peer switches from a segment to the next one.

Finally, SplitStream [72], as the name suggests, splits a live stream into several flows or strips which are transmitted over different trees. As the burden of transmission over a tree lies on the intermediate nodes and the authors aim at creating a fair system, each peer needs to be an intermediate node in at most one tree and a leaf node in the others. In this case, this set of trees is said to be *interior-node disjoint*. Multiple Description Coding (MDC) [73] – a technique based on SVC in which the more separate and individually decodable sub-streams or descriptions of a video are received, the better its final quality will be – can be used in this system, however, as far as the negative points of this solution are concerned, tree-based architectures are usually more expensive to maintain with respect to mesh.

4.3.2 Multimedia Streaming Using BitTorrent and BiToS

The solution proposed by [74] for multimedia streaming using BitTorrent introduces a new selection policy for both chunks and neighbours. Regarding chunks, a sliding window like the one shown in Figure 4.2 is used to separate w pieces that follow the current playback position from the rest of the stream. In this case, w is the length of the window, which is strictly correlated to playback delay. Pieces that do not belong to this window are not downloaded. Pieces that follow the last chunk in the window might be downloaded in the future, but are not considered interesting at present, as their playback position is still too far away. Missing pieces that precede the playback position are not of any use anymore and therefore they will be ignored. Inside the window, chunks are downloaded according to a rarest-first policy. In fact, they might be more appealing to peers that download the video both in a sequential-like way and in the standard BitTorrent fashion. As far as neighbour selection is concerned, modifications are made

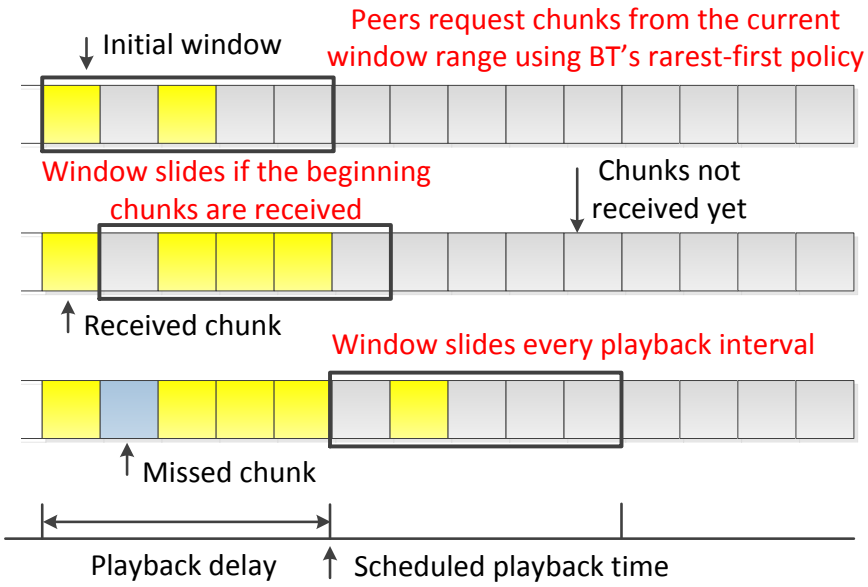


Figure 4.2: Sliding window notion using the rarest-first policy for BitTorrent.

to achieve a better performance during the start-up phase. A randomised version of T4T policy, which "gives more free tries to a larger number of peers in the swarm", is used to give an advantage to those peers that have got nothing or little that they can share.

Another solution, BitTorrent Streaming (BiToS) [75], uses a "High Priority Set", which is similar to a sliding window and pieces are picked from this or from the "Remaining Pieces Set" with probabilities of p and $1-p$ respectively. The optimal choice of p indicates that there exists a trade-off between requesting pieces that are urgently needed for playback and rare pieces that are appealing to other peers. This is one of the key problems in BitTorrent multimedia applications.

Both of these techniques were considered innovative at the time they were proposed (2006-2007). In fact, they are an attempt to use BitTorrent in applications it was not originally intended for. However, they also show some limitations. For example, they do not consider that many peers might have different playback positions and therefore they might be mainly interested in small and different subsets of the original sequence, which is an issue that will be tackled by the approach presented in the next section.

4.3.3 Tribler BitTorrent Client

Tribler [68] is a BitTorrent client that was developed at Delft University of Technology, in the Netherlands. It is based on ABC [Yet Another BitTorrent Client] [76], which is itself based on BitTornado [77]. New features include exploitation of social relationships among peers, which will be described in Chapter 6, and content discovery through exploration of the network instead of browsing a torrent repository.

Give-to-Get Algorithm

The typical course of action of a peer in this system is the following: when it starts downloading a video, playback is delayed until the end of the pre-buffering period. This happens when the estimated downloading time required to complete the sequence is shorter than its playback time. After this, playback starts. When it finishes playing, the peer leaves the swarm; if the download has finished before the end of the playback, that peer will act as a seeder for a certain amount of time. Within this interval of time, the behaviour of the peer can be described by the *give-to-get* (G2G) algorithm [9], which consists of three parts: an unchoking algorithm, which is the module responsible for resource allocation, a piece picking and a neighbour selection algorithm. An important observation is that here seeking and fast-forward are not taken into account, and are not even supported by G2G, even though in the future the protocol might be extended.

Unchoking Algorithm The video stream is divided into smaller pieces, as it happens with any other file. Since G2G does not know anything about codecs, different GOPs might belong to the same piece and vice-versa. Each peer shares information about the pieces it has with its neighbours, which might send requests for them. These neighbours can be unchoked according to their performance evaluation or optimistically unchoked. Typically, at least 3 and at most $3+n$ neighbours, where n could be for example equal to 2, are unchoked according to their rank until 90% of the upload bandwidth is saturated, while at most one peer is optimistically unchoked. It is obvious that if the number of requests is less or equal to $n+4$, all the peers will be unchoked. A pseudo-code version of the proposed method is given in Algorithm 4.3.1. Peers are ranked according to the number of pieces they have forwarded in the last δ seconds, hence the "give-to-get"

name, and at the end of this period the best peers are unchoked; optimistic unchoking, instead, is performed every 2δ seconds.

Algorithm 4.3.1: GIVETOGET(N^*)

choke all neighbours N^*

$N \leftarrow$ all interested neighbours

sort N on forwarding rank

for $i \leftarrow 1$ **to** $\min(|N|, 3+n)$

do
- unchoke ($N[i]$)
- $b \leftarrow \sum_{k=1}^{i}$(our upload speed to $N[k]$)
- **if** $i \geq 3$ **and** $b >$ UPLINK $\cdot\, 0.9$
  - **then break**

The ranking procedure is made of two different steps: first of all "children" peers c are sorted decreasingly by a "parent" peer p according to the number of chunks they downloaded from it that have been forwarded to other "grandchildren" peers g. Second, ties are broken according to the total number of pieces the children have forwarded to their grandchildren. Both steps are required to create a fair mechanism. In fact, assuming that only the pieces received from p and uploaded to g are considered, it will be extremely difficult for a peer c that has been optimistically unchoked to become a good uploader of those chunks over a short period of time. Therefore, also considering the total number of forwarded chunks helps in identifying well-behaving peers. On the other hand, if only the total number of forwarded pieces is considered, a few peers with high capacity will be unchoked by a large number of neighbours and the others will be left to starve.

Another issue is evaluating the number of pieces that have been forwarded, since it is obvious that a parent cannot trust its children c. Therefore, a parent peer p will not ask them for information about the pieces they have uploaded, but for a list of their children g, which will be contacted in order to have a feedback, as it is shown in Figure 4.3. This way, a peer has no interest in lying about its children and has an incentive to share as much as it can. Free-riders can still download from other peers, since they can be

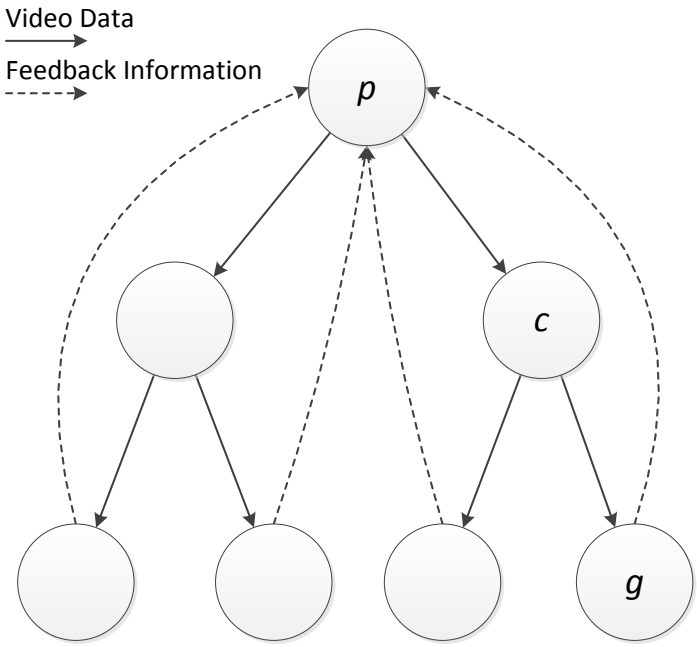


Figure 4.3: The feedback connections for an individual peer.

optimistically unchoked, but if there is no more spare capacity in the system, they will be automatically cut out.

Piece Picking It was already pointed out that in VoD applications for P2P the order which the pieces are downloaded in is crucial, as well as delivering these pieces in time. Therefore, each piece is given a deadline, which is infinite if the video is not being played. A peer is interested in requesting a chunk if three conditions hold: another neighbour has got that chunk, that peer does not have the chunk and has not requested it in the past, and finally there is a chance that it arrives before its deadline.

As far as the piece picking policy is concerned, two important issues have to be taken into account: chunks should be delivered to the video player in order, and therefore should be also downloaded in a sequential fashion, and it is necessary to own pieces that other peers want, in order to get good upload rates. The solution proposed in G2G algorithm gives different priorities to different chunks. Let us suppose that m is the playback position of peer p, where m can be piece 0, if the playback has not started; p will choose the first interesting chunk i in the following three sets:

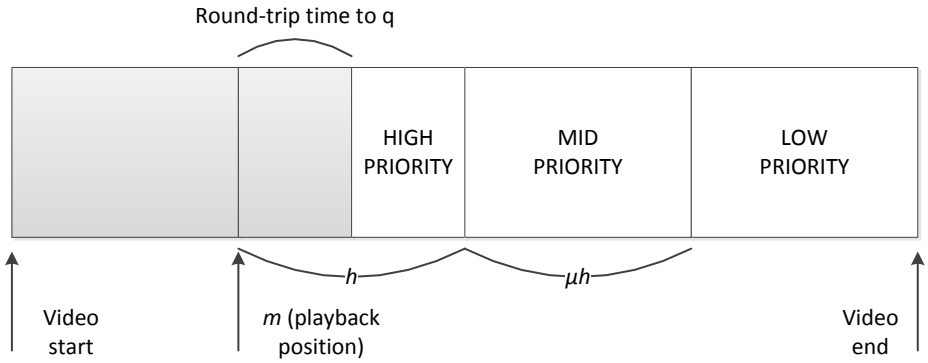


Figure 4.4: Different priority sets in G2G algorithm; the high-, mid- and low-priority sets in relation to the playback position. The chunks in the grey areas, if requested, will not arrive before their deadline.

- **High priority:** $m \leq i < m + h$. If playback has already started, the piece with the smaller index will be downloaded, otherwise the policy will be rarest-first.
- **Mid priority:** $m + h \leq i < m + (\mu + 1)h$. Chunk i will be picked on a rarest-first basis.
- **Low priority:** $m + (\mu + 1)h \leq i$. Chunk i will be picked on a rarest-first basis.

Since mid and low priority are separated, pieces with low priority will be picked only if there are no medium priority chunks left; this distinction is made because mid priority chunks could be part of the high priority set in the near future. If two pieces are equally rare, one of them will be chosen at random. These three sets are shown in Figure 4.4. Finally, every time a new piece is requested, the system estimates the arrival time of all the requested chunks. If this is greater than the deadline, the corresponding request will be dropped. Periodically, the playing buffer is filled with the received pieces and the playback position, which determines the starting point of the different priority sets, is incremented.

Neighbour Selection Tribler also implements a neighbour selection policy. Its aim is to discriminate between "good" or "reliable" peers and "bad" ones. While downloading a video, a peer might fail to deliver a piece belonging to the high priority set before it is required for playing, forcing the system to pause. Therefore, a "black list" of all the peers that have failed to deliver on time is kept (for each download). If the number

of failures exceeds a certain threshold, no pieces from the high or mid priority set will be requested from that peer anymore and they will be picked from the low-priority set instead. This threshold is 4 for live streaming, while it is 0 for VoD.

4.3.4 P2P Scalable Streaming Using a Sliding Window

A few techniques for scalable video transmission over a P2P network have already been proposed. The framework described in [78] can be classified as a "mesh-pull live video streaming system with a T4T strategy". The structure of the overlay network in this case is similar to the original BitTorrent. Advantages of this approach include simple design and robustness to sudden changes in the network, typical of these systems [79]. An important remark is that, while tree-push systems have been extensively studied in Academia, most of practical implementations (like CoolStreaming [80] or the already-cited PPLive [69]) rely on a mesh-pull architecture.

In the considered system [78], a source encodes a video that consists of L layers and each of them is divided into Layer Chunks (LCs). These LCs correspond to Δ seconds of the original sequence. They might have different sizes, depending on the layer they belong to. If a peer wants to watch a video, it requests a list of peers currently downloading it to some entity (whose role is similar to the BitTorrent tracker) and establishes a connection with several of them. In this system, peers exchange information about the chunks they own with their neighbours.

As far as the resource allocation module is concerned, the main idea is that a peer should provide a larger portion of its upload capacity to those peers with the highest download rates. Several receivers might be sending requests to a supplier at the same time, while the supplier only serves one request at a time. The probability $p_{n,k}$ of supplier n serving a request from a receiver k is given in Eq. (4.1):

$$p_{n,k} = \frac{I_{n,k} d_{n,k}^{\gamma}}{\sum_{i \in \mathcal{K}_n} I_{n,i} d_{n,i}^{\gamma}}, \tag{4.1}$$

where $\mathcal{K}_n$ is the set of neighbours of n and $d_{n,k}$ is the current transfer rate from peer k to n. $I_{n,k}$ is defined as 0 if no requests from peer k are in the queue and 1 otherwise. The denominator in Eq. (4.1) thus represents the total transfer rate to peer n from the peers

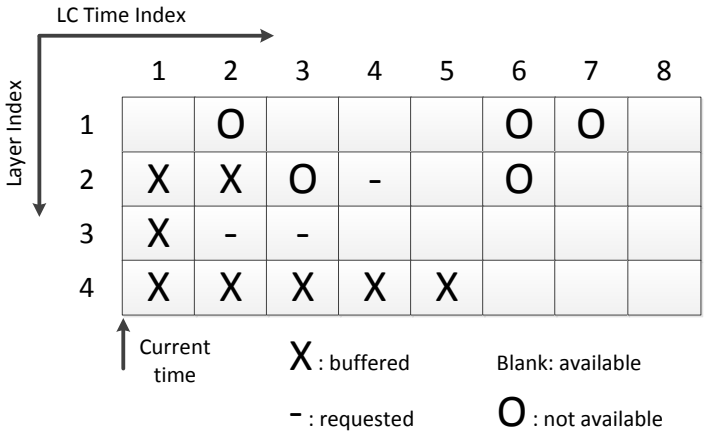


Figure 4.5: Buffer state at a given time.

that have pending requests. Finally, γ is a correction factor. Therefore, $p_{n,k}$ can also be seen as a local reputation, since the peer that uploads the largest amount of data to a supplier has the highest probability of being served. On the other hand, it is clear that free-riding is only tolerated if there is enough spare capacity in the system, for example if there are no other pending requests.

On the receiver side, the scheduler uses a sliding window, which is formed by a fixed number of LCs and video layers. Chunks request is divided into several rounds. At the beginning of each round, the system state is characterised by a buffer, as shown in Figure 4.5. In order to decide which LCs should be downloaded, each of them is assigned a score. This value depends on layer index, playback deadline and rarity. For a generic peer n, the score function for a LC can be expressed as:

$$S_{l,t}^{n} = G(l, t, \lambda), \tag{4.2}$$

where λ represents the number of neighbours of n that currently own a specific piece and l and t are the layer index ad the time index of LC respectively. More specifically, this function can be defined as a linear combination of these three characteristics of a LC:

$$S_{l,t}^{n} = w_1 \frac{l}{L} + w_2 \frac{t - t_0}{B} + w_3 \frac{\lambda}{H}, \tag{4.3}$$

where t_0 is the current playback position, B is the window length, k is the number of neighbours of n, w_1, w_2 and w_3 are weight factors and L has already been defined as the total number of layers. LCs with the highest scores are requested first. These are LCs belonging to low quality layers, with a time index that is close to t_0 and are owned by a limited number of neighbours. An important remark is that only pieces which are believed to be received on time are requested. This estimation is based on the performance of a peer's neighbours. Finally, in order to completely exploit upload and download capacities of peers, low-priority requests are defined.

The idea of using a sliding window has been extensively applied in P2P video streaming, and this solution is no exception. However, in this architecture no extra importance is given to the base layer of the sequence, as the weight factors w_1, w_2 and w_3 are fixed.

4.4 Proposed Approach

The block diagram of our proposed solution is shown in Figure 4.6. It highlights that our contributions described here lie in the chunk selection mechanism, and more specifically in the piece picking and neighbour selection policies. Our piece picking policy formulates how the scalable layers [32] of WSVC video are prioritised. It associates different gains, which result in different priorities, to different parts of the bit-stream, in order to allow video playback while the sequence is being downloaded. In practice, this policy selects the subset of the video bit-stream with the highest bit-rate that can be currently afforded by a user. Moreover, the desired behaviour of the system is that playback should never pause. Pauses occur when a user fails to receive the base layer of a GOP before it is needed for decoding. This may happen even if the overall download bandwidth is high, if these chunks are requested from slow peers. Therefore, our neighbour selection policy is applied. It does not allow a user to request this part of the bit-stream from peers that might fail to deliver it in time. As the diagram also shows, our technique uses the original G2G unchoking algorithm [9] and social-based aspects are not considered here yet.

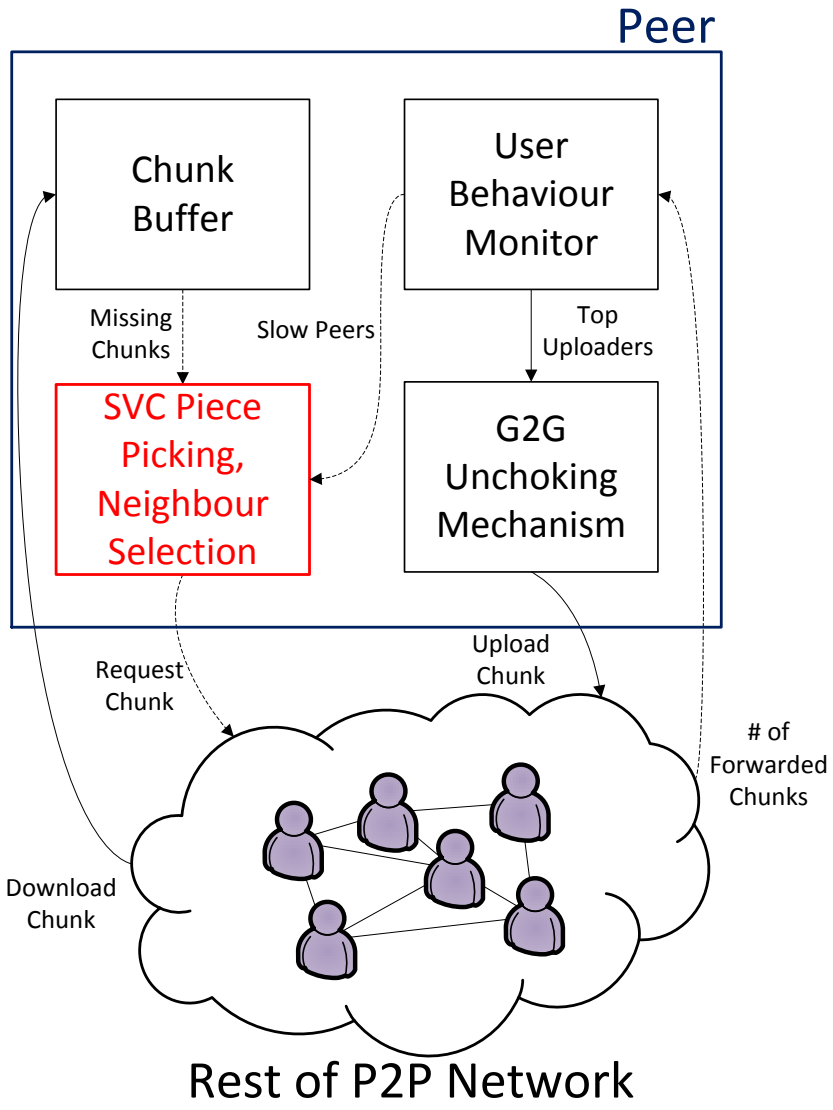


Figure 4.6: Block diagram for the proposed approach.

4.4.1 Piece Picking Policy

This work has also been published in [12, 14, 15]. First of all, WSVC scalable video bit-streams are split into GOPs and atoms as explained in Section 2.3.3. However, only quality scalability is considered here, in order to limit the complexity of our approach. Therefore, groups of atoms will form *quality layers*. An important remark is that atoms can be grouped in different ways in order to create spatial or temporal layers and deciding to exploit another type of scalability does not affect the proposed algorithm. As it will

be shown in the experimental evaluation in Section 4.6, it is also possible to exploit combined scalability by choosing the right encoding parameters.

On the other hand, according to BitTorrent protocol files are split into *chunks* or *pieces* [1]. Since there is no correlation between these two divisions, some information is required to map GOPs and layers into pieces and vice versa. This information can be stored inside an index file, which should be transmitted together with the video sequence. Therefore, the first step consists of creating a new torrent that contains both files. It is clear that the index file should have the highest priority and therefore should be downloaded first. Once the index file is completed, it is opened and information about offsets of different GOPs and layers is read.

For the purpose of explanation, we now assume that all BitTorrent pieces are available from at least one peer. In other words, we assume that there are no *missing pieces*. Moreover, we consider a generic playback position. At this point, it is possible to define the following quantities and functions:

L, total number of GOPs in the sequence.

Q, total number of quality layers.

W, is a window size (in GOPs).

(t, q), represents a GOP index (t) and quality layer index (q), where $(t, q) \in (0, ..., L) \times (0, ..., Q)$. Each of these ordered pairs is associated to a set of BitTorrent pieces.

t_p, current playback position (GOP). Since playback has already started, $t_p > 0$.

$g(t, q)$, gain function. It measures the gain currently associated to BitTorrent pieces of (t, q).

$r(t, q)$, request function. It returns the number of BitTorrent pieces associated to (t, q) that have never been requested.

S is the set of possible candidates $((t_1, q_1), \ldots, (t_N, q_N))$ which a new picked BitTorrent piece belongs to.

$P(S) = (p_1, \ldots, p_k)$ is a function that maps a set of WSVC GOPs and layers S into a set of BitTorrent pieces; $p_1, \ldots, p_k$ are sorted according to their piece id.

This policy uses a sliding window, which consists of W GOPs. The first GOP in the window is the one with index t_p+1, while the last one has index t_p+W. Assuming that $L \gg t_p + W$, the following relations hold:

$$\begin{aligned} g(t,q) &= 0 \\ \text{if } \ & (t \leq t_p, q \in (0,\ldots,Q)) \vee r(t,q) = 0, \end{aligned} \tag{4.4}$$

$$\begin{aligned} g(t,q) &= \frac{1}{1+q} \\ \text{if } \ & t \in (t_p+1,\ldots,t_p+W), q \in (0,\ldots,Q), \end{aligned} \tag{4.5}$$

$$\begin{aligned} g(t,q) &= \frac{1}{1+Q+(t-(t_p+W))} \\ \text{if } \ & t \in (t_p+W+1,\ldots,L), q \in (0,\ldots,Q). \end{aligned} \tag{4.6}$$

First of all, Eq. (4.4) indicates that no pieces lying before the window will be requested, as no gain is associated to the corresponding (t,q) pair. The same holds for any (t,q) whose pieces have already been requested. Eq. (4.5) indicates that pieces inside the sliding window that correspond to the same quality layer also have the same gain. Finally, Eq. (4.6) shows that pieces lying after the window have a gain that only depends on the GOP they belong to. Moreover, these pieces always have a smaller gain with respect to those inside the window. An important remark is that the gain function in Eq. (4.5) and Eq. (4.6) has been defined in such way as it is a monotone decreasing function and could be replaced with any other function that satisfies the same property.

Therefore, a peer requests pieces from layer $i+1$ only if it has already requested layer i completely. Similarly, a peer only requests pieces lying after the window if all the pieces inside the window are already marked as requested. As defining priorities in such way aims to achieve a uniform received video bit-rate among the GOPs inside the window, this mechanism also allows to reduce fluctuations in the received video quality, which results in smoother transitions. This offers a better subjective quality to the final user, as indicated by the results in Chapter 3. That is, this mechanism limits the impact of either positive or negative peaks in the download bandwidth, provided that these variations occur over a period of time that is shorter than the duration of the sliding

window. Finally, as a longer window results in a longer permanence of a GOP inside it, increasing its size makes the system less vulnerable to bandwidth fluctuations, which results in a smoother video playback. However, this comes at the expenses of a longer delay in case of live streaming and a longer pre-buffering time in VoD.

The piece picking rule consists of two steps:

1. Finding the best set of candidates S, which is given by $S = \arg\max_{(t,q)} \{g(t,q)\}$. It is a set of (t_i, q_i) that corresponds to a set of BitTorrent pieces $P(S) = (p_1, \ldots, p_k)$.
2. Choosing a piece from this set. At this step, the decision rule is the following. Given $S = ((t_1, q_1), \ldots, (t_N, q_N))$:
   - If $q_1 = \cdots = q_i = \cdots = q_N = 0$, picked piece p will be the one in $P(S)$ with the smallest id that has not been requested yet. These are pieces that belong to the base layer. This rule gives a higher priority to the base layer of GOPs according to their distance from the current playback position, since the closest ones are the most urgently required for avoiding pauses, as we will see.
   - Otherwise, the rarest piece in $P(S)$ that has not been already requested will be picked. This is the standard BitTorrent picking policy applied to a smaller set.

The window shifts every Δ_{GOP} seconds, where Δ_{GOP} represents the duration of a GOP. The only exception is given by the first shift, which is performed after the pre-buffering, which lasts $W \cdot \Delta_{GOP}$ seconds. Pre-buffering only starts when the index file has been received. Every time the window shifts, two operations are made. First, downloaded pieces are checked, in order to evaluate which layers have been completely downloaded. Second, all outstanding requests regarding pieces of a GOP that lie before the window are cancelled. An important remark is that the window only shifts if at least the base layer has been received, otherwise the system will auto-pause. As far as missing pieces are concerned, they are treated as chunks that have already been requested. However, this only happens during the piece picking phase. If a missing piece belongs to the base layer, the system is paused until it is received correctly. Otherwise, the available chunks will be extracted and decoded.

Figure 4.7 shows the behaviour of the system with $W = 3$. An early stage of the pre-buffering phase is shown in Figure 4.7a. The peer is downloading pieces from the

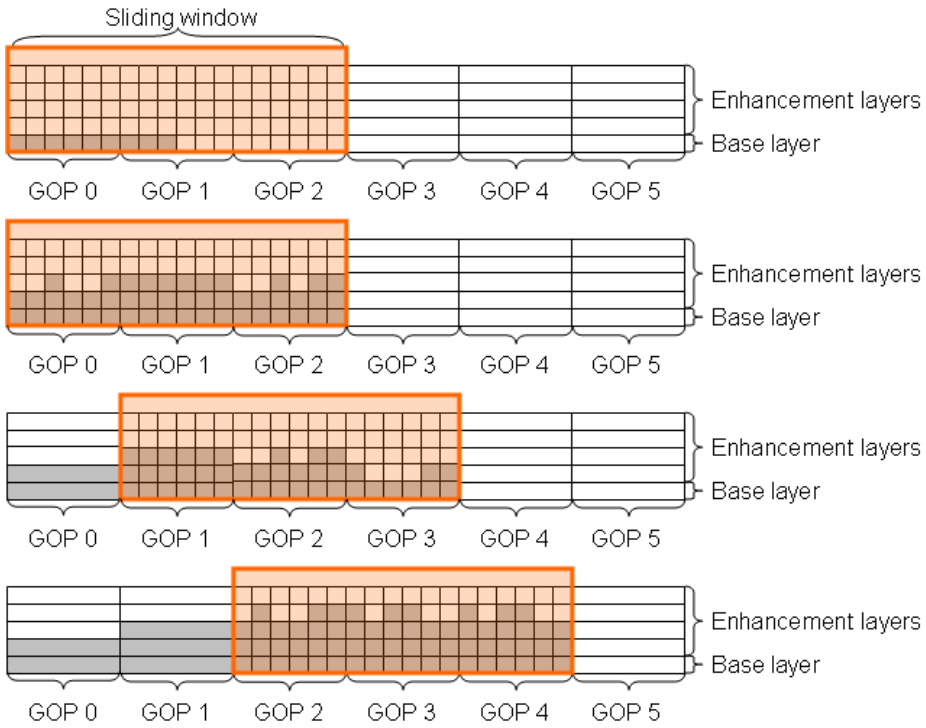


Figure 4.7: Sliding window for scalable video bit-stream a) Pre-buffering phase starts, b) Pre-buffering phase ends, c) The window shifts after GOP 0, d) The window shifts after GOP 1.

base layer according to their piece id, while in Figure 4.7b the first two layers have been downloaded and pieces are being picked from the enhancement layer 2 according to a rarest-first policy. In Figure 4.7c, the window has shifted. As not all the pieces of enhancement layer 2 of GOP 0 have been received, this layer and higher layers are discarded. In this phase, pieces from the base layer and the enhancement layer 1 of GOP 3 have a higher priority with respect to enhancement layer 2 of GOP 1 and 2. In Figure 4.7d all the GOPs have the same number of complete layers and pieces are picked from enhancement layer 3.

An alternative approach which exploits FGS of WSVC could use only two layers (base and enhancement). The base layer could correspond to a video which has the minimum acceptable quality according to QoE-related considerations. On the other hand, the algorithm could dynamically set a target received video bit-rate depending on

the current network conditions and request all the chunks belonging to this enhancement layer according to a rarest-first policy. This would increase the chance of downloading data that cannot be decoded – due to the hierarchical structure of WSVC bit-stream – but also the probability that the set of chunks different peers own is more diverse, which is one of the keys to efficient resource allocation.

An important remark is that this solution presents some analogies with [78], illustrated in Section 4.3.4. However, a WSVC video is used in our work. Moreover, the correlation between LCs and BitTorrent chunks is not considered and the base layer is not downloaded sequentially. Another difference is the neighbour selection policy, which is illustrated in the next section.

4.4.2 Neighbour Selection Policy

It is extremely important that at least the base layer of each GOP is received before the window shifts. Occasionally, slow pieces in the swarm might delay the receiving of a BitTorrent piece, even if the overall download bandwidth is high. This problem is critical if the requested piece belongs to the base layer, as it might force the playback to pause. Therefore, these pieces should be requested from good neighbours. Good neighbours are those peers that own the piece with the highest download rates, which alone could provide the current peer with a transfer rate that is above a certain threshold and this set is updated at every base layer chunk request, as it depends on the set of peers that own that chunk. During the pre-buffering phase, any piece can be requested from any peer. However, every time the window shifts, the current download rates of all the neighbours are evaluated and the peers are sorted in descending order.

Let us suppose that p is a piece belonging to the base layer that a peer wants to download and $N_1, \ldots, N_k$ are the peer's neighbours that are currently uploading to (or *unchoking* [1]) it that own this piece. Since neighbours are sorted, $R(N_1) > \cdots > R(N_i) > \cdots > R(N_k)$, where $R(N_i)$ indicates the current download rate from neighbour i and k is the number of neighbours. The threshold value is calculated as:

$$R_T = \frac{n_0 \cdot l(p)}{W \cdot \Delta_{GOP}}, \tag{4.7}$$

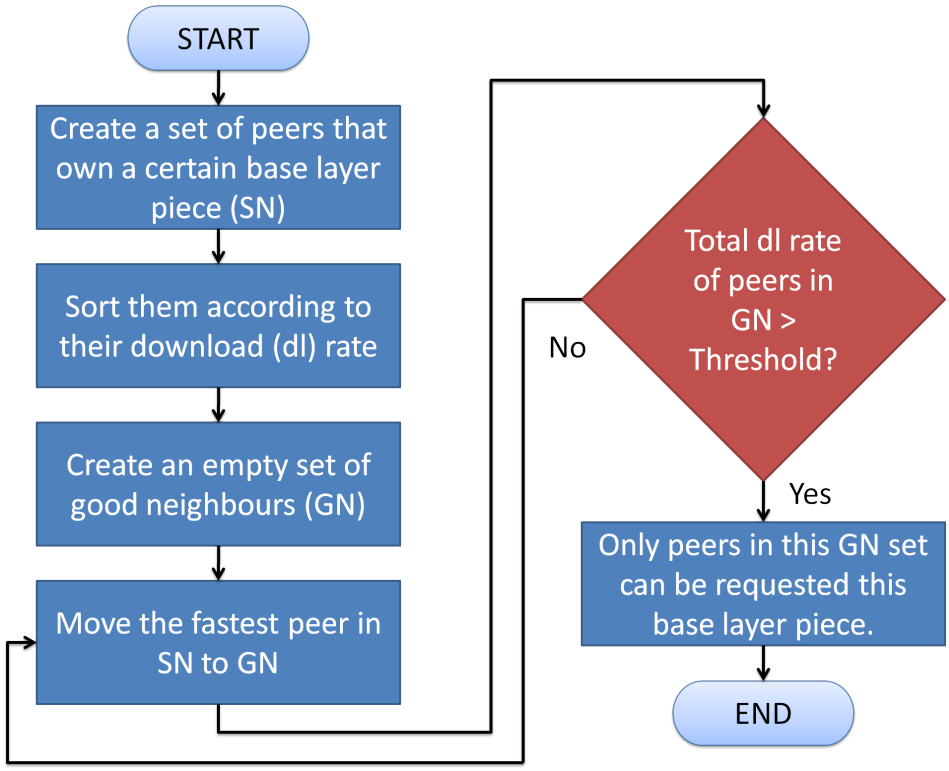


Figure 4.8: Flowchart diagram of our proposed neighbour selection policy.

where n_0 represents the number of pieces in the base layer that are currently inside the window, $l(p)$ is the size of the BitTorrent piece for this file and the other quantities have been already defined. In other words, R_T is the minimum rate that allows these pieces from the base layer to be received in time. Assuming that

$$T^* \hat{=} \min_T \sum_{i=1}^{T} R(N_i) > R_T, \quad T \leq k. \tag{4.8}$$

T^* is the minimum number of neighbours which own this piece, whose sum of download rates exceeds the threshold and $N_1, \ldots, N_{T^*}$ is the set of neighbours from which base layer pieces can be requested. A flowchart diagram explaining this algorithm is shown in Figure 4.8.

An important remark is that, as far as BitTorrent is concerned, it has been proven that peers with similar upload bandwidths tend to connect with each other [81], and the P2P network becomes clusterised; that is, high bandwidth hosts are matched with

other high bandwidth hosts, and low bandwidth hosts are matched together. This is also known as *stratification*. It is out of the scope of this thesis to prove whether this phenomenon also appears when G2G is applied, like in this case, or for different resource allocation algorithms, however, our neighbour selection policy can in principle be used in BitTorrent when a peer's good neighbours have not been discovered yet.

4.5 Implementation

In this section we briefly describe how we created the tools to test our algorithms. The starting point was Tribler BitTorrent client, which has already been described in Section 4.3.3. This client has been written using Python [82] programming language. The part of the program that handles Video on Demand and Live Streaming consists of several modules, as shown in Figure 4.9. The main components in this section are:

- **VideoStatus**: This module handles information about the current video sequence and keeps track of the playback position. It has knowledge of the position of the sequence inside the torrent and also implements the window used in the G2G algorithm.
- **PiecePickerVOD**: Inside the source code, it is implemented as *PiecePickerStreaming*. It implements the piece picking policy of G2G, which means that it has a knowledge of the current playback position and the download window implemented in VideoStatus. Therefore, it avoids requesting obsolete pieces. This module is derived from PiecePicker, which keeps track of all the pieces owned by all the peers in the swarm and sorts them according to their rarity.
- **MovieOnDemandTransport**: This module was implemented as *VideoOn DemandTransport* in the source code. It provides an interface between BitTorrent and a video player. It can start the playback, pause it, resume it and it can push available data into the player buffer. This counts as playing the video, as far as Tribler is concerned, and in practice consists in incrementing the playback position.
- **StorageWrapper**: It handles the received data and has methods that allow to read it.

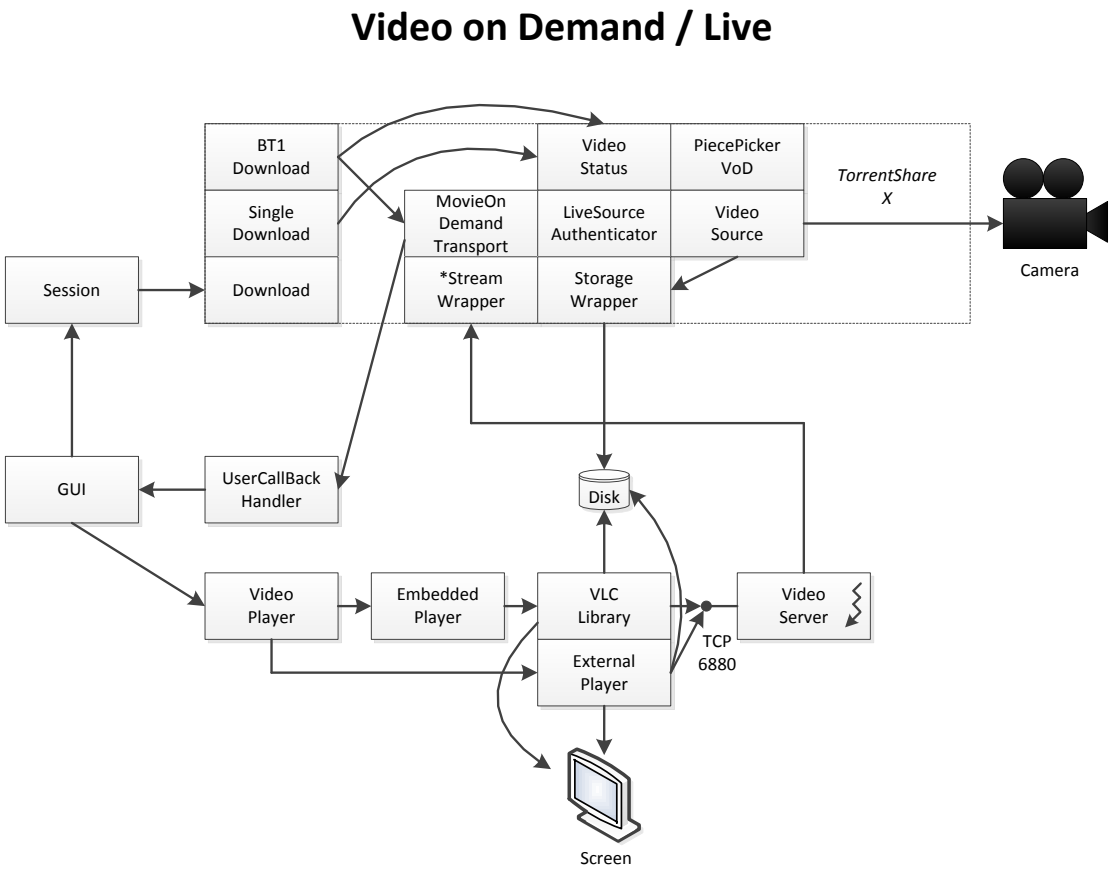


Figure 4.9: Tribler Video Architecture.

- **VideoSource** and **LiveSourceAuth**: These modules are used for live streaming. In this case we cannot create a torrent that contains a hash for all the pieces, since we do not have all the data when we start broadcasting. VideoSource handles the live source and injects data into the network, while LiveSourceAuth is used to create a hash of every single piece and check the integrity of the received data.

Some of these modules have been modified, while a few others have been created. Figure 4.10 shows the main changes that were introduced. Implementation of the proposed algorithms and their integration with the rest of the system required over 1000 lines of code written in Python [82]. The new piece picking policy has been implemented in *PiecePickerStreaming*. It has a method that selects the next piece to pick, which simply implements the algorithm described in Section 4.4.1. It also contains the neighbour selection policy. This method is called every time we are trying to request a piece from any of our neighbours. If we have selected a piece that belongs to the base layer,

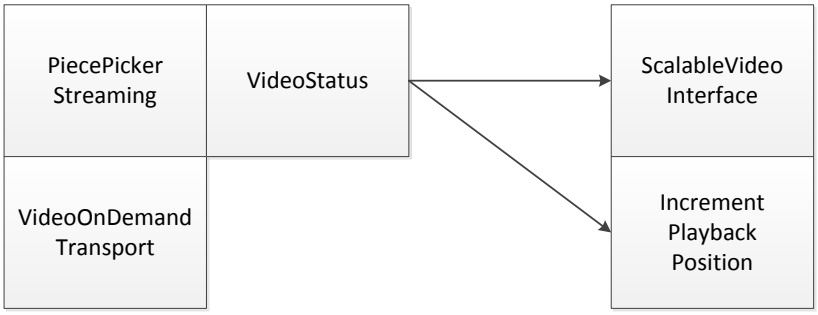


Figure 4.10: New modules used for scalable video streaming.

we check if the current peer belongs to the "good peers" set. If it does, we go on with the request, otherwise we put this piece in an "off-limits" list and we request another piece. This list is required because some pieces might belong to the base layer and one or more other layers. Extraction is performed inside *VideoOnDemandTransport*, while we defined a new module, *IncrementPlaybackPosition*, which is responsible for the window shifts. The interface that makes Tribler "codec-aware" is also implemented in a new module, *ScalableVideoInterface*.

4.6 Results

4.6.1 Dataset Description

For our experiments, first of all four uncompressed video sequences have been generated. All of them consist of the repetition of a ten-second YUV video sequence, which can be "City", "Crew", "Soccer" or "InToTree". These are commonly used sequences in testing video compression and they cover different types of content in terms of spatial and temporal activity. All the YUV sequences that were processed by the encoder had CIF frame resolution (352×288 pixels) and a frame rate of 30 fps. The duration of all sequences was 300 seconds.

Original sequences are shown in Figure 4.11. These have been subsequently encoded with the parameters shown in Table 4.1, where extraction points have been specified. An important remark is that compressed videos have multiple spatial and temporal

YUV Sequence	GOP Size	Extraction Points [bps]	
City	128 frames	CIF, 30Hz QCIF, 15Hz	384k, 448k, 512k, 640k, 768k 96k, 112k, 128k, 160k, 192k
Crew	64 frames	CIF, 30Hz	192k, 224k, 256k, 320k, 384k, 480k, 576k, 672k, 768k, 1024k, 1280k, 1536k
Soccer	64 frames	CIF, 30Hz QCIF, 15Hz	384k, 448k, 512k, 640k, 768k 96k, 112k, 128k, 160k, 192k
InToTree	64 frames	CIF, 30Hz	192k, 256k, 320k, 480k, 576k, 640k, 768k

Table 4.1: Encoding parameters and extraction points.

resolution extraction points, which means that low quality layers only contain a version of the video with lower frame size and frame rate, and therefore we are *de facto* only exploiting one type of scalability. This can be also seen in Table 4.1. On the other hand, in the encoded version of Crew and InToTree only quality can be degraded. The specified size for BitTorrent pieces is 32 kB.

As far as the P2P network used for testing is concerned, it consists of three peers. Two of them are seeders and one is a leecher. Depending on the experiment, they are allowed to leave and re-join the network or not and, moreover, different limits on the upload bandwidth of each peer are set. More details about specific upload capacities can be found in Section 4.6.2 and Section 4.6.3, where the individual experimental set-ups are described.

4.6.2 Comparison Between a Non-scalable and a Scalable Codec

In this section, we will compare the performance achieved by WSVC and a non-scalable video codec. The behaviour of a non-scalable codec is however simulated by handling a scalable sequence. In order to obtain a non-scalable video, the window described in Section 4.4.1 only shifts if all the quality layers have been received. The sequence selected for this experiment is Crew. The sliding window consists of 6 GOPs, for a total duration

(a) City

(b) Crew

(c) Soccer

(d) InToTree

Figure 4.11: Original City, Crew, Soccer and InToTree sequences.

of 12.8 s. As far as network configuration is concerned, as we previously mentioned there are two seeders (Peer 1 and Peer 2) and one leecher (Peer 3). Upload bandwidth limit has been set to 25 kB/s and 50 kB/s respectively, for a total of 600 kbps. Therefore, this bandwidth is not enough to download the full-quality video, which has a bit-rate of 1536kbps. Moreover, Peer 1 periodically disconnects and reconnects to the network, which causes fluctuations in the download bandwidth of Peer 3.

Figure 4.12 and Figure 4.13 show the download rate and the received video bit-rate in this case. The received video bit-rate here needs to be intended as the playback bit-rate and a received rate of 0 kbps means that the system is being paused due to buffering. Some extracted video frames for the scalable sequence are reported in Figure 4.14 (b)-

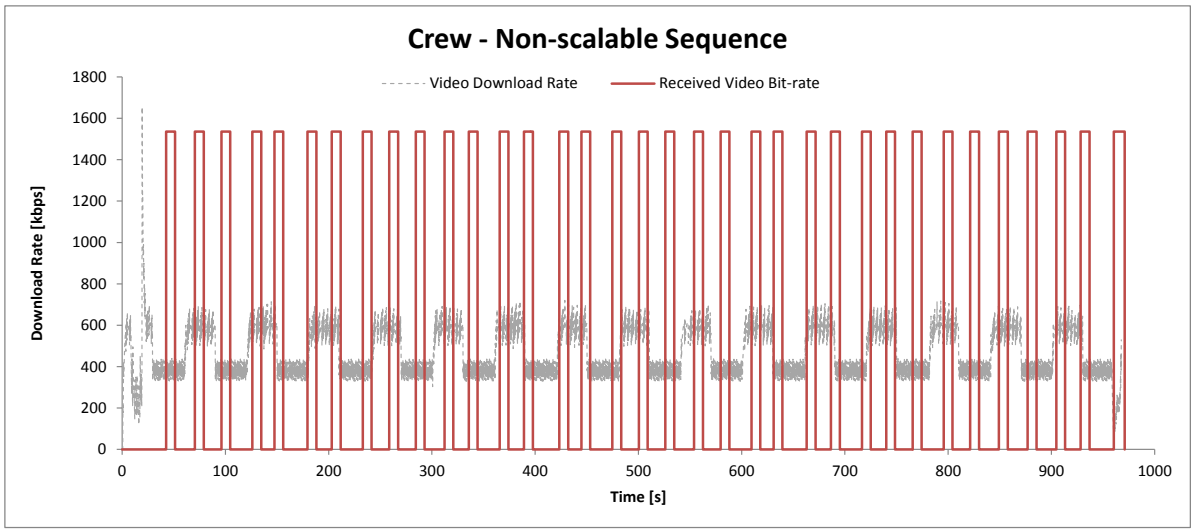


Figure 4.12: Received video bit-rate for Crew, non-scalable case.

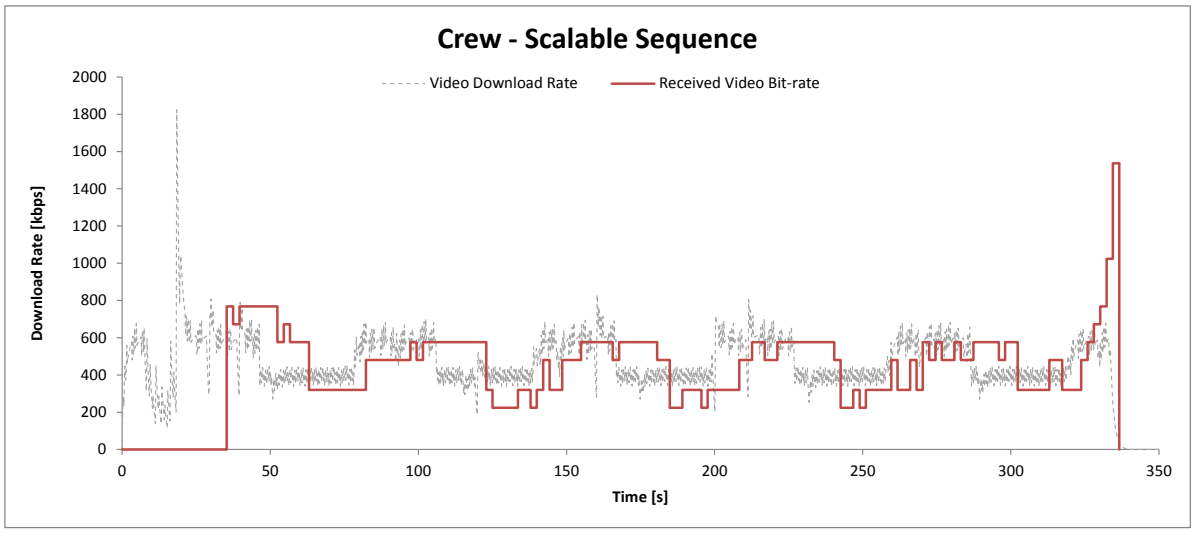


Figure 4.13: Received video bit-rate for Crew, scalable case.

(d), while the original one is shown in (a). The graphs show that in the non-scalable case, since the overall download bandwidth is low, the system needs to pause several times during the playback. On the other hand, when using a scalable video codec, the received video quality follows the download rate, as the variations occur over a period of time (about 30 s) that is longer than the sliding window (12.8 s). However, variations do not occur immediately, but after a while. It is important to note that that the received video bit-rate is in general a little lower than the download bandwidth. One of the reasons is

(a) Original Frame

(b) Frame 8012, Video Bit-rate = 256 kbps PSNR = 34.29 dB

(c) Frame 2312, Video Bit-rate = 576 kbps PSNR = 36.40 dB

(d) Frame 8912, Video Bit-rate = 1024 kbps, PSNR = 38.78 dB

Figure 4.14: Example of decoded frames with different qualities.

that layers may be split on different BitTorrent pieces and BitTorrent pieces can contain more than one layer. This problem could be solved by exploiting FGS of WSVC. In this example, we can observe that the received video quality increases towards the end of the sequence. This happens because the window shrinks when there are no GOPs left to replace those that have already been decoded. As far as decoded frames are concerned, in the scalable case a received video bit-rate of 576 kbps still leads to acceptable results (Figure 4.14 (c)) . However, when it drops to 256 kbps, or even less, quality is very low (Figure 4.14 (b)). A possible remark is that this comparison might not look fair, as a non-scalable video sequence with a lower bit-rate could allow a user to receive the video

in time without pausing. However, this experiment models all those cases in which the video bit-rate is fixed and the download bandwidth is either temporarily or permanently too low to sustain this video bit-rate and what we would like to highlight with this test is the additional flexibility deriving from the SVC approach.

4.6.3 Effects of Neighbour Selection Policy

In this set of experiments, the effect of the neighbour selection algorithm is evaluated. The impact of this policy is more evident if at least one slow peer is in the swarm, as it can delay the completion of a piece, which could belong for example to the base layer. The window size chosen for these tests is $t_W = 8.53$ s. As in the previous tests, the network consists of two seeders and one leecher, however, there are one "fast" and one "slow" peer, whose upload limit has been set to 40 kB/s and 3 kB/s respectively. The latter is the minimum acceptable value that can be set in Tribler. The total upload bandwidth is therefore 344 kbps. Tests have been performed on City, Crew, Soccer and InToTree sequences both when neighbour selection policy was active and when it was not.

Results are shown in Figure 4.15 to 4.22 and in Table 4.2. As far as City and Crew are concerned, playback is paused only a few times when neighbour selection is not active, while Soccer and InToTree perform very badly. On the other hand, when this policy is active, playback never stops. As far as the average video bit-rate is concerned, City and Crew show very similar performances with and without neighbour selection, while Soccer and InToTree show a higher received video bit-rate, since the videos have been downloaded for a much longer time.

According to these results, this policy actually helps in delivering the base layer in time. However, most of the graphs show drops in received bit-rate, especially Figure 4.16 and Figure 4.20. These suggest that, even if our neighbour selection is active, pieces from low quality layers might not be received on time, since protection is only given to the base layer. Therefore, in order to achieve a smoother playback quality, a more complex policy might be required.

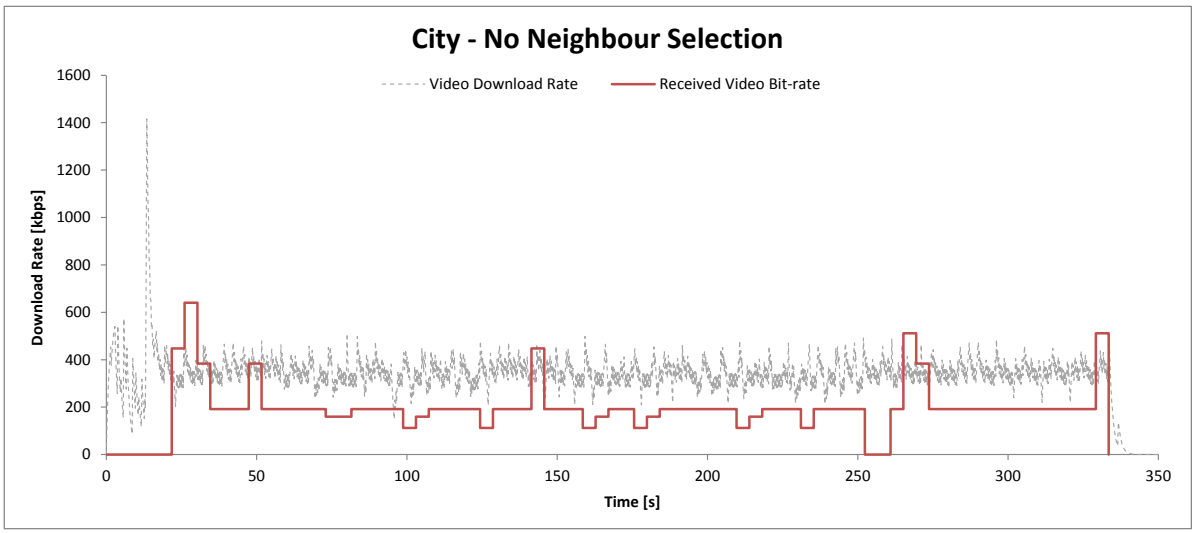


Figure 4.15: Received video bit-rate for City. Neighbour selection is not active.

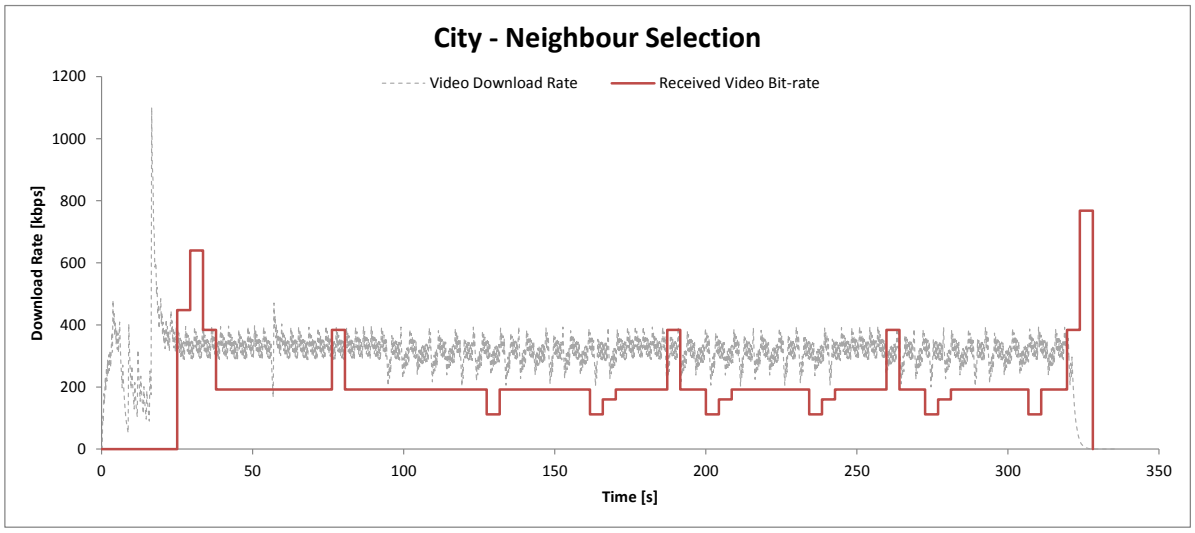


Figure 4.16: Received video bit-rate for City. Neighbour selection is active.

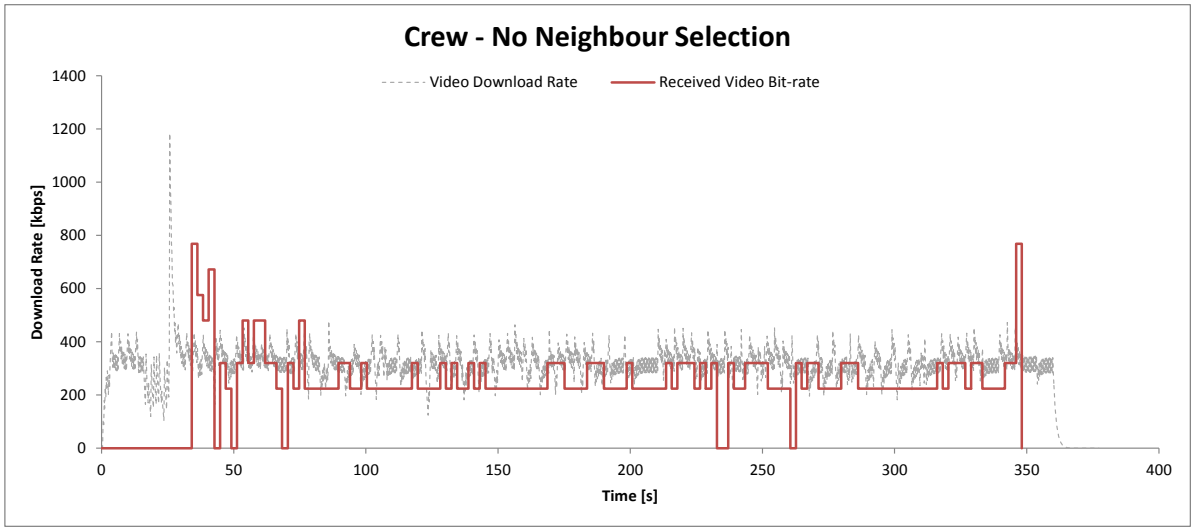


Figure 4.17: Received video bit-rate for Crew. Neighbour selection is not active.

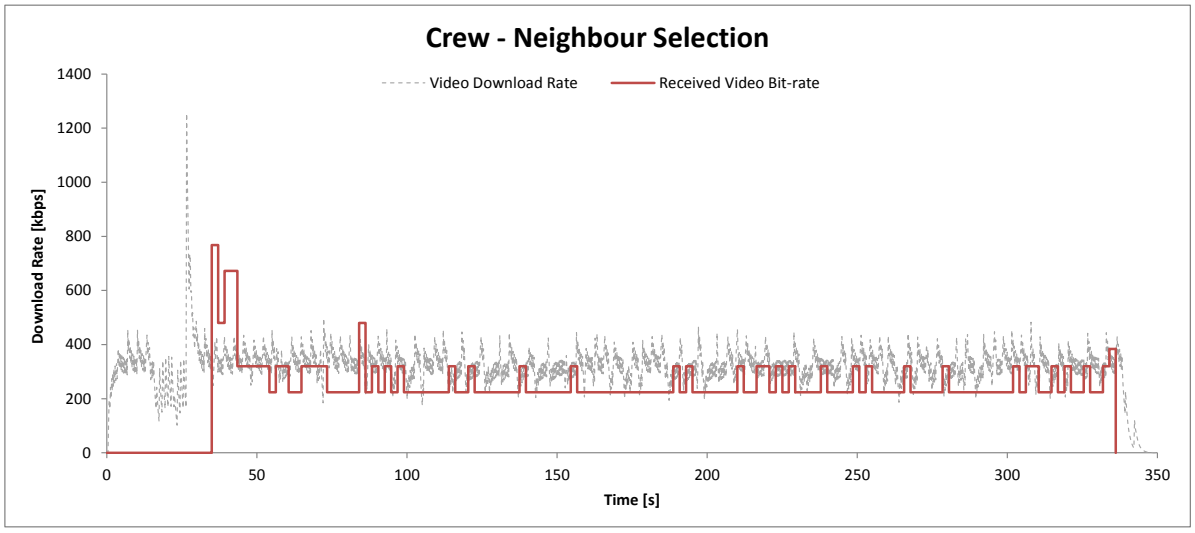


Figure 4.18: Received video bit-rate for Crew. Neighbour selection is active.

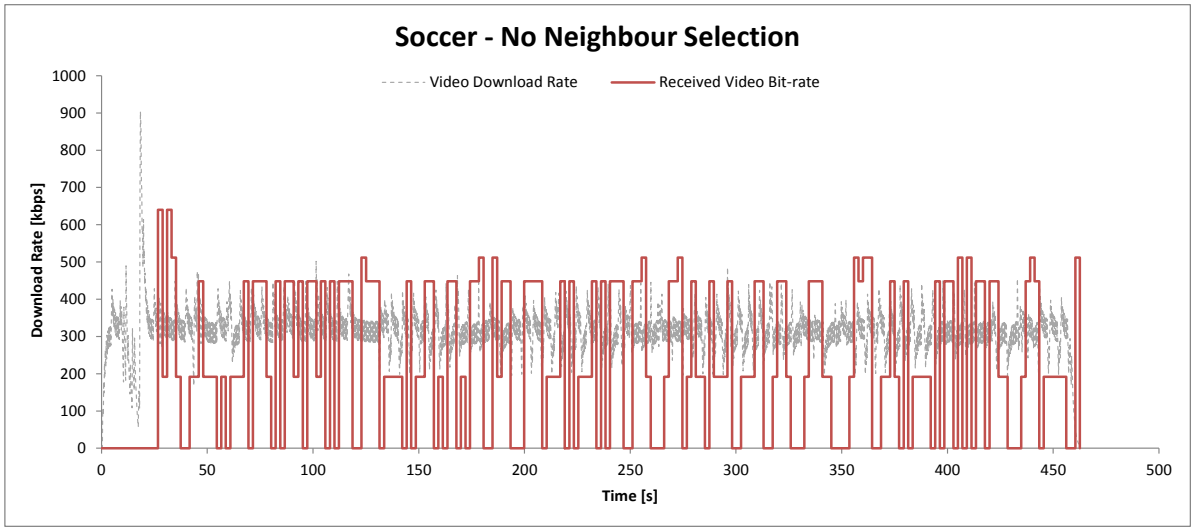


Figure 4.19: Received video bit-rate for Soccer. Neighbour selection is not active.

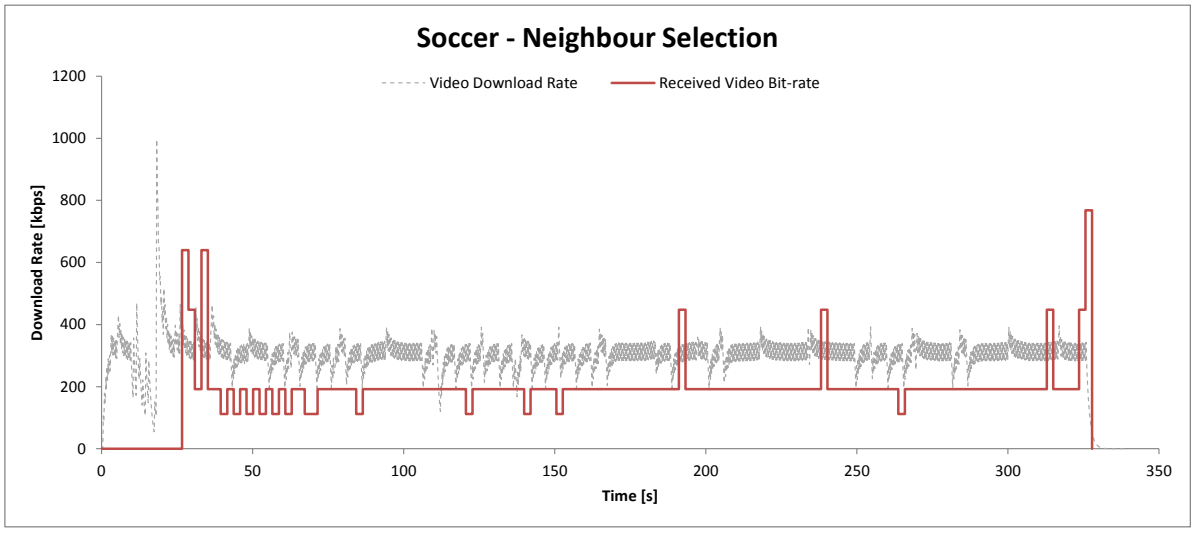


Figure 4.20: Received video bit-rate for Soccer. Neighbour selection is active.

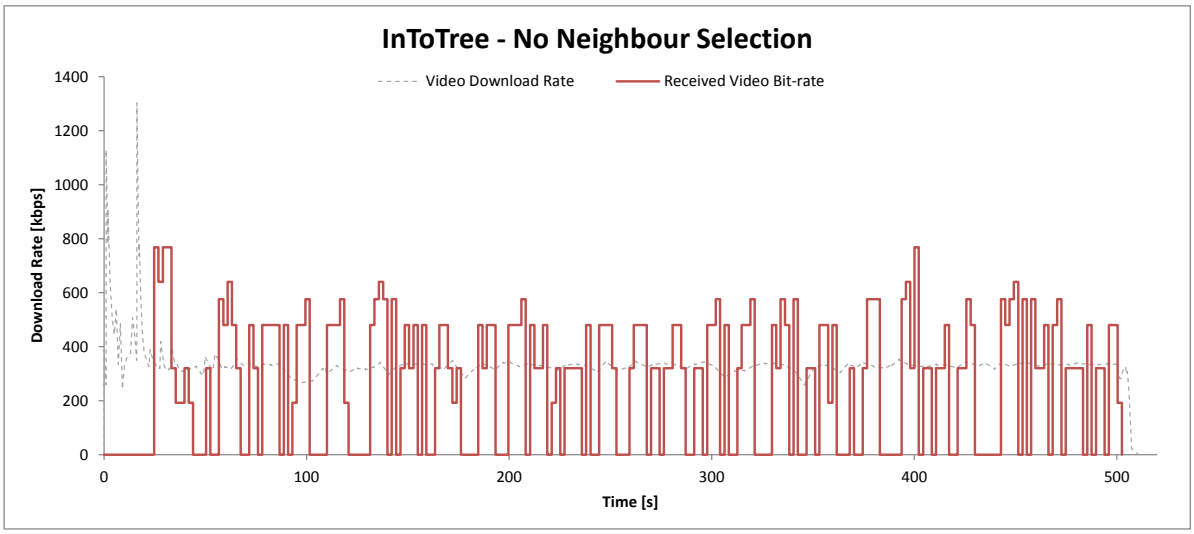


Figure 4.21: Received video bit-rate for InToTree. Neighbour selection is not active.

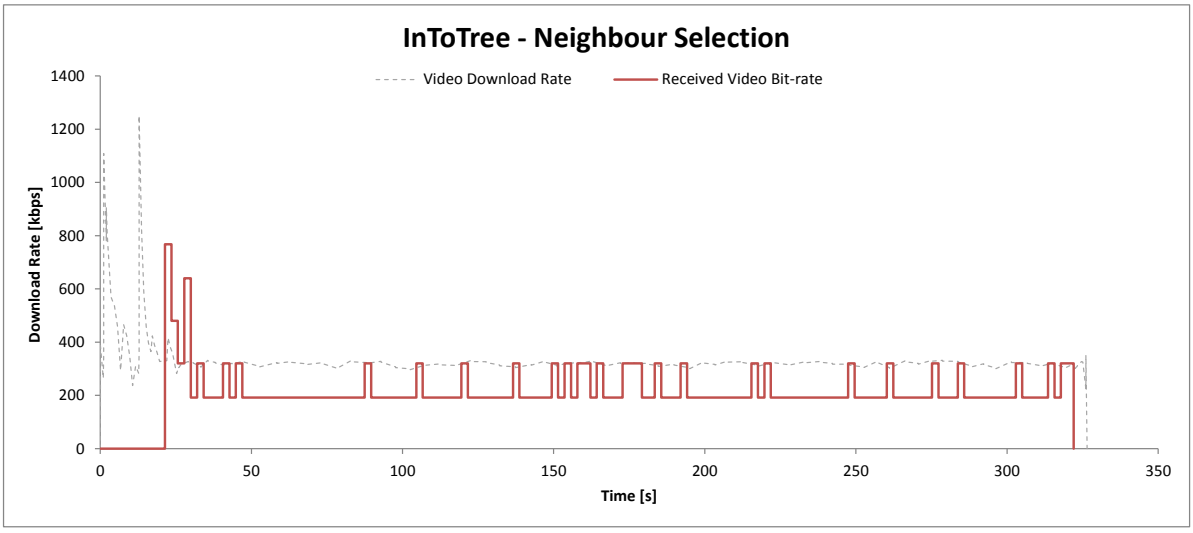


Figure 4.22: Received video bit-rate for InToTree. Neighbour selection is active.

Another important remark is that the testing environment used for these experiments is too limited. However, as performing real tests on a much larger-scale environment might be very expensive, the most reasonable option is to use a simulation tool.

Sequence	Average bit-rate	Autopauses (duration)
City w/o N.S.	213 kbps	2 (8.53 s)
City w/ N.S.	215 kbps	0
Crew w/o N.S.	278 kbps	6 (12.8 s)
Crew w/ N.S.	264 kbps	0
Soccer w/o N.S.	345 kbps	64 (136.5 s)
Soccer w/ N.S.	204 kbps	0
InToTree w/o N.S.	435 kbps	82 (174.7 s)
InToTree w/ N.S.	227 kbps	0

Table 4.2: Average received bit-rates with and without neighbour selection policy.

4.7 Conclusion

First of all, in Figure 4.1 we showed a general diagram which can be taken as a reference for the technique explained in this chapter and in the following ones. Moreover, we presented two algorithms that allowed efficient streaming of scalable video sequences over a P2P network. Experimental evaluation showed that our piece picking policy allowed adaptation of the received video bit-rate to the available resources through accurate chunk selection. Due to the hierarchical fashion of quality layer priorities, this policy also aimed at providing a received video quality that was as constant as possible, in accordance to the conclusions that were drawn from the subjective video analysis in Chapter 3. Moreover, our neighbour selection policy significantly reduced or eliminated playback pauses by only requesting base layer pieces to reliable peers. The strongest aspect of this technique is its compatibility with the original BitTorrent. That is, this solution could be easily adopted on a large scale and this would be transparent to the other peers in the swarm, provided that in the future more sequences will be encoded in WSVC format. Alternatively, it would be straightforward to adapt this solution to the standard H.264/SVC. On the other hand, these algorithms do not take network

resources optimisation into account, for example in terms of maximising throughput. Moreover, while G2G algorithm is suitable for Video on Demand, as it assumes that peers have different playback position, a T4T-like unchoking mechanism is necessary in case of live streaming. However, in this context peers might not be able to reciprocate resources immediately, therefore a different solution is needed. In the next chapter, we will propose a different approach specifically tackling this problem.

Chapter 5

Joint Free-riding Detection and Resource Optimisation

This chapter describes a credit-based framework for scalable video transmission over a P2P network, where the goal is to cut out misbehaving users and optimise resource allocation at the same time. This is an evolution of our previous approach, described in Chapter 4. This model is based on direct resource reciprocation, however, time constraints are more relaxed with respect to BitTorrent. Therefore, we lose compatibility with the original protocol. For this contribution, we are adopting the following approach: first of all we consider a real P2P video transmission case and try to identify how this problem – too difficult to solve analytically – can be simplified introducing constraints, even if unrealistic. This leads to a much easier problem, for which we propose a solution. Some of the considerations that are valid for the simplified case are useful when we try to remove some of these constraints and our proposed approach itself is based on some of these hypotheses. This chapter is organised as follows: Section 5.1 introduces the problem; Section 5.2 describes a very simple case study, which will be useful to deal with a more complex problem; Section 5.3 discusses the relevant related techniques; Section 5.4 illustrates the proposed approach; Section 5.5 briefly discusses how the experimental evaluation framework has been implemented; Section 5.6 presents the simulation results; and finally, Section 5.7 concludes the chapter by analysing the pros and cons of this approach.

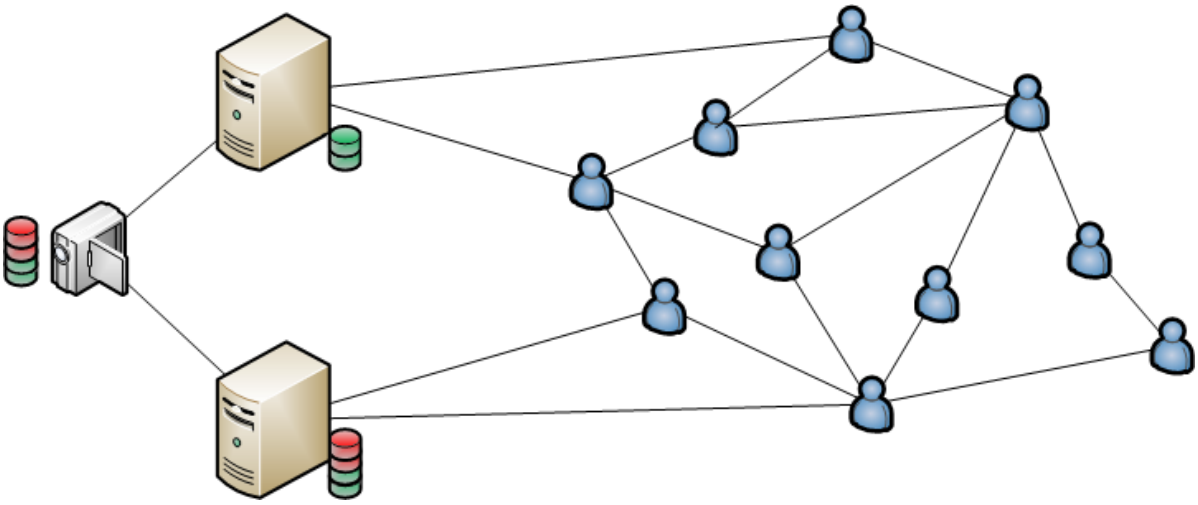


Figure 5.1: A P2P TV scenario.

5.1 Problem Description

In our previous approach, we did not consider the problem of resource reciprocation, fairness and optimal resource allocation. In order to achieve a P2P system for scalable video streaming that addresses these aspects, our research objectives are now to:

- Propose an algorithm that identifies free-riders and isolates them from the rest of the system, to reward well-behaving users.
- At the same time, optimise resource allocation. This is to be intended as maximising the usage of the upload capacities of all the peers in the network (that intend to cooperate).

First of all, we illustrate a realistic case and analyse its related issues, then we introduce some hypotheses that simplify the system and study a reduced version of the problem.

In a real scenario, we can suppose to transmit a scalable video sequence over a BitTorrent-like P2P network. The context is now real-time streaming. The video is composed of two quality layers, corresponding to different bit-rates. The base layer has a bit-rate of 256 kbps, while this layer plus the (only) enhancement layer have a bit-rate of 512 kbps. Since the size of the P2P chunk is 32 kB and the source encodes one GOP every two seconds, in this time interval two base layer pieces and two enhancement layer

pieces will be generated. The network consists of $N = 100$ peers, which have an upload bandwidth of 64 kB/s plus two sources, used to inject the content into the network. One of them shares the base layer of the sequence only, while the other uploads both layers to the remaining peers. These two sources have an upload capacity of 256 kB/s and 128 kB/s respectively, which is not enough to upload the sequence to all the peers. This can be however achieved using the resources of the other users of the network. Figure 5.1 illustrates this scenario with 10 peers. If we want to solve the problem of optimal resource allocation in this scenario, it is not a simple tasks, as there are several degrees of freedom: what is the strategy the source nodes should follow? how many users should they upload data to at the same time? what is the best strategy the other peers should adopt? In order to find a solution to these issues, we will first study a simplified version of this scenario.

5.2 A Simplified Case Study

Given the scenario introduced in the previous section, it is possible to identify the following hypotheses, which simplify the problem we are trying to solve. Most of them are unrealistic and have very restrictive constraints, however more degrees of freedom will be added later on:

- There is only one source injecting content into the network.
- There are no constraints on the download capacity of the peers.
- All the peers have the same upload capacity.
- The video is not scalable.
- All the peers know the current state of the network, e.g. who their neighbours are what are the pieces owned by them.
- The network is static. No peers are allowed to join or leave the network.
- There are no node failures and the network latency is always zero.
- All the peers wish to cooperate, e.g. there are no free-riders or users that inject polluted content into the network.

- The nodes are strongly connected; that is, there is a path from every node to every other node in the network.

The aim of **each individual peer** is to receive each GOP of the sequence within a certain time. On the other hand, the objective of the **service provider** is that all the peers receive the video sequence in time and their playback position is approximately the same. In other words, peers should be synchronised.

5.2.1 An Analytical Solution

Under these circumstances, we would like to find the best resources allocation profile. This is the profile that allows all the peers to receive a certain GOP within the minimum possible time. According to the original BitTorrent protocol, a peer can start uploading a piece only after it has been completed downloaded (and verified). In other words, a piece is the minimum unit of content that can be shared (even if made of sub-pieces [1]). In order to use the resources of a peer as soon as possible, this peer needs to have something to share as soon as possible. If we consider a sequence that consists of only one GOP, consisting of only one piece, intuitively the allocation that allows it to be received in the shortest time by all peers is the one in which all the peers unchoke[1] the same user. In fact, when only the source owns the piece, if it only unchokes one peer, this peer completes the download of the piece in the shortest possible time. When this happens, both peers should unchoke a new peer (the same) and so on. Since we assumed that the source and the peers have the same upload capacity, if we consider that the time required to transfer a piece from the source to the first unchoked peer is τ, it can be proven (for example by induction) that the time δ_N required to transfer this GOP to N peers is given by:

$$\delta_N = \tau + \frac{\tau}{2} + \cdots + \frac{\tau}{N} = \tau \sum_{n=1}^{N} \frac{1}{n}. \tag{5.1}$$

For $\tau = 1$, this is the sum of the first N terms of the harmonic series [83]. Because of the properties of this series, if $N \to \infty$, then $\delta_N \to \infty$. Therefore, if we set a maximum acceptable threshold for the playback delay, there is a maximum number of users that

[1]We remind that, according to BitTorrent jargon, unchoking means uploading data to a certain user.

can be part of the system. This happens even in this unrealistic scenario where there is no limit on the download bandwidth and we do not even consider which parts of the piece are downloaded from which peer. If the source generates a GOP at regular intervals, each peer should forward the last piece they received. Similarly, all the peers that are uploading the same piece should unchoke the same peer. This way, all the peers are kept constantly busy and new pieces are made available for upload as quickly as possible. In this case, the number of peers that are currently uploading a piece is limited and it depends on the interval between two GOPs. Therefore, also in this case we have a maximum number of users that can take advantage of the service within a (set) acceptable delay.

It is not realistic to consider GOPs that are made by only one piece. We will now consider the propagation of a one-GOP sequence that is made of M pieces. In this case, the problem of allocation becomes much more complicated. First, we will show an example where $M = 2$, then we will consider more generic values for it. Let us now suppose that the original GOP is divided into two pieces. The source now unchokes two peers at the same time, uploading two different pieces. When the download is complete, the scenario becomes similar to the previous example. However, in the next phase, peers owning different pieces should unchoke different peers. Moreover, they should unchoke peers that do not already own the other piece. This is because this peer's resources would be already fully exploited and the resources of another peer would not be used. When all the peers own something to share, they need to upload it to other peers. The strategy used at this point is irrelevant, as regardless of the number of peers that a peer unchokes or the number of the peers that are transmitting the same piece to another user, the time required to upload a piece to all the other users is the same. Therefore, in this case, we have an "initialisation" phase, during which all peers need to obtain something to share and another phase in which all peers exchange data. Moreover, the duration of the second phase, under these circumstances, does not depend on the number of peers in the network. An example with $N = 10$ shows that $\delta_N = 2.9289\ \tau$ if $M = 1$ and $\delta_N = 2.2873\ \tau$ if $M = 2$. Therefore, dividing a GOP into smaller parts reduces the maximum delay.

If we now consider a network with N peers and $M = N$, we can observe that the source transmits $1/N$ of the GOP to N peers in τ seconds. All the peers now have something to share and they can upload it to the other users. Even if we do not consider

the source, which may be transmitting the following GOP, the time required to upload this part of the GOP to the others is $((N-1)/N)\tau < \tau$. Therefore, this system is virtually scalable and it can be used to transmit videos if the interval between two GOPs is greater than τ.

In the original BitTorrent protocol, the number of chunks per GOP depends on the video bit-rate, as the piece length is fixed. In realistic scenarios, we have that $M \ll N$. Assuming that N/M is an integer value for simplicity, we consider two possible initialisations:

- The source unchokes all the peers at the same time: in this case, the initialisation phase has the duration expressed by Eq. (5.2), while the exchange phase is calculated as $t_{ex} = ((M-1)/M) \cdot \tau$.

$$t_{init} = \tau \frac{N}{M}. \tag{5.2}$$

- The source unchokes one peer per piece and all the peers that own the same piece upload it to the same user. In this case, the initialisation time is calculated in Eq. (5.3), while the exchange phase has the same duration as before.

$$t_{init} = \tau \left(1 + \sum_{n=1}^{(N/M)-1} \frac{1}{1+Mn} \right). \tag{5.3}$$

In both cases, the initialisation phase duration depends on the number of peers in the network, while the second one does not. However, it depends on the overall upload capacity of the peers (as it would have an influence on τ). Table 5.1 shows different results for the initialisation time as computed by Eq. (5.3), depending on the number of chunks and peers. It is clear that dividing a GOP into a larger number of parts leads, theoretically, to better performances.

These scenarios are not realistic, as many of the constraints they are based on are not. However, some of the ideas that emerged from them could still be valid in more generic cases. The aim of this study is not to find a comprehensive solution to the

	$M = 3$	$M = 9$	$M = 27$	$M = 81$	$M = 243$
$N = 3$	τ	—	—	—	—
$N = 9$	$1.3928\ \tau$	τ	—	—	—
$N = 27$	$1.7703\ \tau$	$1.1526\ \tau$	τ	—	—
$N = 81$	$2.1405\ \tau$	$1.2846\ \tau$	$1.0538\ \tau$	τ	—
$N = 243$	$2.5081\ \tau$	$1.4099\ \tau$	$1.0986\ \tau$	$1.0183\ \tau$	τ

Table 5.1: Initialisation times for different values of M and N.

considered problem. It should rather raise questions and suggest ideas that will be used both in our proposed approach and in our future work.

For example, one key aspect is that in order to keep upload channels busy, peers should always have content other users are interested in. In particular, different users need to have different content. In order for this to happen, it is reasonable to upload data to the most deprived peer, as we increase the likelihood that this peer owns content that is interesting for other users. However, this consideration is not valid when there are free-riders in the network. In fact, in order to achieve fairness, the amount of data each user receives should roughly correspond to the amount of data it uploads. These two aspects are not related, and this is an issue we will consider in our technique.

5.2.2 Additional Problems

Let us now analyse the constraints we previously introduced. Some of them will be removed when proposing our approaches, whilst the others might provide ideas for future work, as discussed in Chapter 7.

There is more than one source injecting content into the network If we consider a scalable video, they might have different versions of the sequence. These sources should work in parallel and be coordinated by some authority.

There are constraints on the maximum download bandwidth of the peers This factor is not normally considered, as it is not the bottleneck of P2P systems.

However, this implies that there should be a limit on the number of peers unchoking another one, or peers could be allowed to unchoke more than one peer at a time (to reduce the transfer rates towards them).

Peers have different upload capacities This aspect invalidates the nice symmetry and synchronisation properties that we observed in the simple case. First, we could consider a discrete set of upload capacities, then we could allow this parameter to be completely random.

The video is scalable Dividing GOPs into pieces of the same length might not be a good idea. Moreover, this strict rule was the main reason that pushed us to exploit only one type of scalability in our work so far. Moreover, scalability could be also used to "synchronise" peers with different upload or download capacities.

Peers do not know the current state of the network Peers in general have a very limited knowledge of the network. Every decision needs to be based on this. Some concepts like friendship or reputation have been already introduced in other systems, but they can be extended.

Peers are allowed to join or leave the network The system should adapt to these changes and ideally this problem should be transparent to the other users.

The network is unreliable Original BitTorrent uses TCP. However, the User Datagram Protocol (UDP) has been used in P2P systems. Moreover, peers might have different response latencies.

There are free-riders and malicious peers Not all the users wish to cooperate. Another issue is that it is sometimes difficult to distinguish between a peer with a bad connection and a free-rider. Credit-based models usually see P2P networks as markets and define rules that make systems resilient with respect to these attacks. Improvements might be also given by applying game theory.

Peers need to establish a connection with a limited number of users Creating the overlay network can be an issue. Original BitTorrent is based on centralised trackers, while distributed ones already exist. The creation of such network could be based on friendship relationships. Properties of these networks, like topology, should be taken into account.

5.3 Related Work

Before proceeding to the description of our proposed approach, we are going to present a few related techniques. They are divided into two groups. First, we are going to analyse some frameworks for resource reciprocation in P2P systems, with a particular focus on those that are specifically designed for video transmission, either scalable or non-scalable. An important remark is that, in principle, systems in the latter category could be extended to support scalable video sequences by designing a chunk selection mechanism which is aware of the codec characteristics. Second, we will consider techniques that optimise resource utilisation in P2P networks.

5.3.1 A Simple File-sharing Model

In [84], Buragohain et al. proposed a game theoretic framework for incentives provision in a P2P network. This study was performed in 2003 and therefore it is as old as the original BitTorrent protocol. Such a system is modelled as a *non-cooperative game*, as users compete for shared resources and can decide not to give any contribution. The aim of each player is to maximise their own payoff or utility function, while their level of contribution is the strategy they adopt. The authors focus on the concept of *Nash Equilibrium* [8]; that is, they analyse those cases in which players cannot increase their utility by unilaterally changing their strategy.

The set of players is made of N peers $P_1, \ldots, P_N$ and the utility function associated to peer P_i is U_i. Finally, the contribution given by the ith peer is denoted by D_i. This quantity is a continuous variable and can correspond, for example, to a certain amount of data shared in a fixed period of time and can be also described in terms of costs and therefore money, if multiplied by a cost factor c_i. This contribution can be also

expressed using a dimensionless quantity defined as

$$d_i \equiv \frac{D_i}{D_0}, \tag{5.4}$$

where D_0 is a fixed measure of contribution. The incentive mechanism will aim to ensure that each peer contributes at least D_0 to the system.

Another important feature of this system is the use of a *benefit matrix* B. This is a $N \times N$ matrix, where B_{ij} is the value of the contribution peer P_i made to P_j and it can be also measured in actual money. In general, $B_{ij} \geq 0$ and $B_{ii} = 0 \ \forall i$. The following dimensionless parameters are also introduced:

$$b_{ij} = \frac{B_{ij}}{c_i}, \qquad b_i = \sum_j b_{ij}, \qquad b_{\text{av}} = \frac{1}{N} \sum_i b_i. \tag{5.5}$$

In these equations, b_i is the total benefit that peer i can obtain from the other users of the network and b_{av} is the average benefit for all the peers in the system.

The basic idea behind this model is that if a contribution made by a peer is small, the benefit it gains will also be small. In practice, a peer P_j accepts a request from P_i with a probability $p(d_i)$ and rejects it with probability $1 - p(d_i)$. This probability function should be monotonically increasing, in order to satisfy the properties previously stated. The function used in [84] is defined as follows:

$$p(d) = \frac{d^\alpha}{1 + d^\alpha}, \qquad \alpha > 0. \tag{5.6}$$

It is easy to observe that $p(0) = 0$ and $p(d) \to 1$ for large values of d. For large values of α, this function has a steep step. Therefore, there exists a threshold under which a peer has a high probability of rejection. An important consideration is that acceptance or rejection only depends on the total contribution to the system.

Considering the quantities that have been previously defined, it is possible to define the following utility function u_i for each peer P_i

$$u_i = -d_i + p(d_i) \sum_j b_{ij} d_{ij}, \qquad b_{ii} \equiv 0. \tag{5.7}$$

The first term here indicates that there is a cost associated to joining the system (d_i), which linearly increases with respect to the data contributed by a user i. On the other hand, the benefit given by joining the system depends on how much the other users are willing to cooperate (d_j). It is possible to prove that neither $d_i = 0$ nor $d_i \to \infty$ are optimal strategies.

As far as the two-player game is concerned, Buragohain shows that there exist two Nash Equilibria [8] if the benefit b given by joining the network is above a certain threshold, which depends on the chosen probability function $p(d)$. One of this equilibria denotes a high contribution from both peers and is indicated by d_{hi}, while the other corresponds to a smaller amount of contributions by the users and is denoted by d_{lo}. Experimental results show that d_{hi} is a stable Nash Equilibrium, while d_{lo} is unstable. Therefore, depending on the initial contribution values chosen by the peers they will both converge to the desirable strategy or leave the system empty-handed. These concepts can be extended to the N-player game.

The solution proposed in [84] is important because it shows that in such a framework there exists a threshold in the benefit received by users under which it is not convenient to join the system. On the other hand, a possible limitation is that only pure strategies are considered here, while no mixed strategies are allowed. Moreover, given the definition of probability function p, it appears the single decisions to accept or refuse requests from other peers depend on their total contribution to the system, rather than the benefit directly obtained from them.

5.3.2 Live Multimedia Streaming in a P2P Social Network

The solution proposed in [85] aims at designing incentive mechanisms in the specific context of live video streaming. In this model, the reward depends not only on the availability of data, but also on the quality of the received video. Another characteristic of such a system is the presence of very strict delay constraints, as already mentioned for other BitTorrent-based video streaming systems. Moreover, the P2P network itself is modelled as a social network, where users can cheat to improve their payoff or tamper with the shared data. Social relations among peers (e.g. friendship) are however not considered.

In this system, a video bit-stream of bit-rate B is divided into chunks of length M. This video is originally stored on a server. A tracker-like mechanism is implemented to allow peers to obtain a list of their neighbours. Finally, time is divided into rounds of length τ. First of all, a two-player interaction will be analysed from a game theoretic point of view. The payoff function of the game for one single round is defined as follows:

$$\begin{aligned}\pi_1(a_1, a_2) &= (a_2 P_{21}) g_1 - a_1 c_1 \\ &= (a_2 P_{21}) g_1 - a_1 \frac{M}{W_1 \tau}, \\ \pi_2(a_1, a_2) &= (a_1 P_{12}) g_2 - a_2 c_2 \\ &= (a_1 P_{12}) g_2 - a_2 \frac{M}{W_2 \tau}. \end{aligned} \tag{5.8}$$

These equations contain several terms. First of all, a_i is the action taken by peer i and it is either 1 or 0, depending on the decision of i to cooperate or not respond to the other peer's request. P_{ij} is the probability that a chunk is successfully transferred from i to j, while $g \in [0, 1]$ denotes the gain a certain user receives when receiving a video chunk; it is a subjective measure. W_i is the upload bandwidth of peer i. Finally, c_i is the cost associated to cooperation and it is measured in terms of percentage of bandwidth used to transmit one chunk in one round. There are two terms in the equation. The first one is the gain of user i with respect to the action of the other peer, while the second is the cost, which depends on their own action. If the game is only played for one round, the only action profile (a_1, a_2) that satisfies Nash Equilibrium [8] is $(0, 0)$, similarly to what happens in the game theoretic problem called *prisoner's dilemma* [86]. The same holds if the game is played for a finite number of times and the termination time is known to both users. However, in these systems, users do not exactly know when the other peers will leave the game. Therefore, other Nash Equilibria exist and the game can be modelled as an infinitely repeated game.

In this case, an averaged payoff function over all the rounds of the game $U_i(\mathbf{s})$ is considered, where $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$ is the strategy profile for the two players. For this game, the set of feasible and enforceable payoffs is given by

$$\begin{aligned} V_1 = \text{convex hull } &\{(v_1, v_2) \mid \exists (a_1, a_2) \text{ with } (\pi_1(a_1, a_2), \pi_2(a_1, a_2)) = (v_1, v_2)\} \\ &\cap \{(\pi_1(\mathbf{x}), \pi_2(\mathbf{x})) \mid \pi_1(\mathbf{x}), \pi_2(\mathbf{x}) \geq 0\}. \end{aligned} \tag{5.9}$$

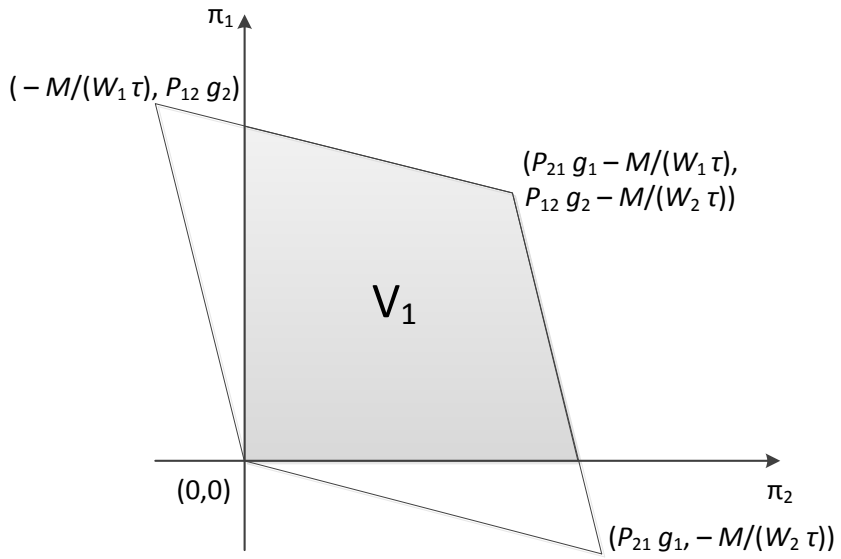


Figure 5.2: Feasible and enforceable payoff profiles.

where $(a_1, a_2) \in \{0, 1\}$, as previously stated and $\mathbf{x} = (x_1, x_2)$ is the set of the average strategies for the infinitely repeated game. These payoff profiles are also shown in Figure 5.2. The average strategy of each player is defined as $x_i = \lim_{t\to\infty} \sum_{j=0}^{T} a_i^{(j)}/T$, where $a_i^{(j)}$ is the strategy adopted by i during round j and T is the total number of rounds. For the infinitely repeated game there exists an infinite number of Nash Equilibrium strategies. However, not all of them are acceptable from the point of view of the users of the social network. Therefore, it is necessary to introduce some more bounds, or refinements of the Nash Equilibrium:

- **Pareto Optimality**: It means that it is not possible to increase the payoff of a peer without decreasing the the others'. This subset corresponds to the two segments on the edge of the convex hull between point $(P_{21}g_1 - M/(W_1\tau), P_{12}g_2 - M/(W_2\tau))$ and the two intersections with axes $\pi_1(\mathbf{x}) = 0$, $\pi_2(\mathbf{x}) = 0$.

- **Proportional Fairness**: A payoff profile that is proportionally fair can be achieved if the product $U_1(s)U_2(s)$ is maximised. This corresponds to maximising $\pi_1(x)\pi_2(x)$ at each round.

- **Absolute Fairness**: This solution is not always Pareto-optimal. However, this concept can be applied on many occasions. According to this criterion, the payoff of every player in the game should be the same, e.g. $\pi_1(x^*) = \pi_2(x^*)$.

It is possible to prove that the solutions found by applying absolute and proportional fairness are not cheat-proof. However, when cheating on their private information, peers can maximise their payoff reporting $P_{ji}g_iWi = M/\tau$. The corresponding strategy profile is $\mathbf{x}^* = (1, 1)$, which means that users always cooperate. The corresponding payoff profile is

$$\mathbf{v} = \left(P_{21}g_1 - \frac{M}{W_1\tau}, P_{12}g_2 - \frac{M}{W_2\tau} \right). \tag{5.10}$$

This forms a Nash Equilibrium, is Pareto optimal and cheat-proof, as it already assumes that peers may be cheating about their private information. Other information peers can cheat on is the list of chunks in their buffer. This reduces the number of requests from other users, while getting rewards from other peers. To avoid this, peers should not upload more chunks than they receive from the other one.

For this two-player game, a cheat-proof strategy is obtained when all the users report false values of their private information. Therefore, transmitting such values is useless. In practice, in the two-player game, peers will upload data to peers they assume to be good enough to receive data in exchange. Moreover, they will not upload more data then they actually download. This is a revisitation of the original BitTorrent unchoking mechanism.

For the multiple user scenario, new challenges are presented, as in bigger networks links between users might sometimes be busy and packets can be delayed or lost. Two types of attacks are also considered: incomplete chunks attack and pollution attacks. Under these circumstances, the challenge is to distinguish between malicious and unintentional misbehaviour. Therefore, a *credit line* is introduced. This is the maximum gap allowed between the number of complete and unpolluted chunks two peers download. It is in practice as a "download threshold", since if a peer tries to cross this line by requesting more data, the user that has received the smallest contribution will reject all of the other peer's requests. This rule is valid for all the relationships between a peer and its neighbours. The aim of this rule is to stimulate cooperation among peers and to add some tolerance when favours cannot be instantly returned. Moreover, this limits the damage free-riders or malicious peers in general can cause to the other users.

Another goal is to detect these malicious users. Assuming that the unsuccessful transmission of a chunk is modelled as a Bernoulli random process [87] and applying the

Central Limit Theorem [88], it is possible to prove that:

$$Cs^{(i)}(j,t) - Cu^{(j)}(i,t)P_{ji} \sim \mathcal{N}\left(0, Cu^{(j)}(i,t)P_{ji}(1-P_{ji})\right), \tag{5.11}$$

where $Cs^{(i)}(j,t)$ is defined as the number of chunks that i successfully receives from j per round and $Cu^{(j)}(i,t)$ is the number of chunks that i has requested to j and j has agreed to transmit by time t. This quantity needs to be multiplied by P_{ji}, which has already been defined as the probability to transmit one chunk from j to i in one round. Assuming that a malicious peer will never send a good or complete chunk, it is possible to set a threshold that corresponds to the maximum false positive malicious users rate depending on this Gaussian distribution.

An important remark stated in this work is that in some cases it is more convenient for a selfish peer to download only from the source, without sharing any data with the others. In this case, it will always refuse to cooperate with other peers. However, it is reasonable to assume the streaming server as often busy. Therefore cooperation among otherwise selfish peers can be enforced. For a selfish peer i which is cooperating with the others, the following strategy is defined:

- Mark all the other peers as selfish.
- Send a request to the peer that has the highest probability of successfully transmitting a chunk within a round and apply the malicious behaviour detection rule to it.
- When receiving a request, accept it if the peer is not marked as malicious and the credit line has not been crossed.

It is possible to prove that in the case of no attacks the strategy described in this section strategy is sub-game perfect, cheat-proof and, if none of the peers stops receiving chunks, strongly Pareto optimal. On the other hand, experimental evaluation shows that malicious users either are detected by the proposed mechanism or can only cause very limited damage.

5.3.3 Overview of Other Techniques Based on Game Theory

Other systems have been proposed [89–94]. In many of them, a utility, payoff or welfare function is defined to measure the benefit gained by a certain user or a group when receiving data from other peers minus the upload costs they incur.

In addition, in [89], Park and van der Schaar consider the problem of content production and sharing together. They also analyse what are the conditions under which it is convenient for a peer to cooperate in a P2P system. Moreover, a few pricing schemes are also proposed. A similar scheme was proposed in [95] in 2006.

In [90, 91] a "foresighted" resource reciprocation model is proposed. The main idea is that peers should upload data to other users considering how this action will impact their behaviour in the future. Moreover, peers do not just decide which users they want to upload to, but also the scale of this contribution. This is achieved using a Markov Decision Process [96] framework. Simulations show a significant improvement in the results, however a practical implementation of this system could be computationally very expensive.

A few more techniques analysed are based on evolutionary game theory [97]. In [92–94], peers try to adapt their strategies to the feedback they get from other users. In particular, in [92] peers autonomously decide if they should keep their current list of neighbours or update it. On the other hand, in [94] peers are allowed to behave cooperatively. Moreover, they can either decide to download only from peers in their own group or act as agents (e.g. "superpeers") and contribute to their group downloading data from outside it.

Finally, some properties of the network (like those described in [42, 98–101]) can, in principle, be exploited in general P2P systems, not just for video streaming. So far, there are examples of systems that consider proximity of nodes [102, 103] or social relationships [104] when deciding which connections should be established between peers. The results presented in these works show that these systems perform better than the traditional ones.

5.3.4 P2P Network as a Graph

The following techniques consider the issue of optimal resource allocation in P2P transmission. We now consider the solution proposed by Huang et al. [105]. The overlay network is modelled as a graph $G = (V, E)$. V represents the set of vertices (peer nodes) and E is the set of links among them in the overlay network. Each of these links $(i, j) \in E$ is associated with a communication delay $d_{(i,j)},\ \forall (i, j) \in E$. These links are considered to be symmetric and to satisfy the triangular inequality (e.g. $d_{(i,j)} + d_{(j,k)} \geq d_{(i,k)}\ \forall i, j, k \in V$). Maximum download and upload capacities are defined for every peer as I_i and O_i and they are measured in terms of units (e.g. minimum P2P chunk size) per second.

The stream is generated by only one source S, while all the other peers $r \in R \subset V$ are receivers. This stream can be either received from the source only, or (indirectly) through the other peers via multiple paths. The rate of this (non-scalable) stream is denoted by s, while f_{ij} is the streaming rate from peer i to peer j. The stream from the source to a receiver r is the *unicast flow* U to r, which may be formed by several *fractional flows* $p \in U$. Each of these flows is associated with a delay $t_p = \sum_{(i,j)\in p} d_{(i,j)}$. The latency of the unicast flow is the maximum latency of the fractional flows $t = \max_{p\in U} t_p$. Given these assumptions and definitions, the problem is formulated as follows:

$$\text{Minimise } t,$$

subject to:

$$\sum_{(i,j)} d_{(i,j)} x^r_{ijm} \leq t \ \text{ with } \ x^r_{ijm} \in \{0, 1\},\ \forall (i, j) \in E,\ \forall r \in R,\ \forall m, \tag{5.12}$$

where x^r_{ijm} is set to 1 only if the path of the *m-th* flow from the source to a receiver r includes the link from i to j. This condition simply indicates that t is the maximum delay of all the flows to all the receivers. In other words,

$$x^r_{ijm} = \lceil f^r_{ijm}/s \rceil\ \forall (i, j) \in E,\ \forall r \in R,\ \forall m. \tag{5.13}$$

Here, f^r_{ijm} is the *m-th* fractional flow rate on (i, j) to r. The maximum upload and download rate of each peer also needs to be considered:

$$\sum_{m=1}^{\mathbf{s}} \sum_{r:r\in R} f^r_{ijm} \leq y_{ij},\ \forall (i,j) \in E \quad , \tag{5.14}$$

$$\sum_{j:(i,j)\in E} y_{ij} \leq O_i,\ \forall i \in V, \tag{5.15}$$

$$\sum_{j:(j,i)\in E} y_{ji} \leq I_i,\ \forall i \in V. \tag{5.16}$$

In Eq. (5.14) to (5.16), y_{ij} is the aggregated flow rate on link (i, j). Over all the incoming (outgoing) connections, the sum of aggregated flows cannot exceed the download (upload) capacity of each node. Here, $\mathbf{s}$ denotes the number of fractional flows, therefore $0 < m \leq \mathbf{s}$. This algorithm achieves a delay which can be approximated by $O(\sqrt{\log(n)})$, where n is the number of peers in the network. This paper works with data flows instead of considering packet exchanges, however, it is relevant to our study, as it states that in order for all the peers to be served, the sum of all the upload rates of all the peers needs to be greater than the sum of all the desired rates, which is a statement we are going to extend for the scalable case.

5.3.5 A Push-based Approach for a Mesh Topology

The problem of minimising delays in video transmission is connected to the problem of optimising streaming rates. In [106], Picconi et al. describe a mesh-based approach which can achieve optimal streaming rates, similarly to tree-based solutions. This is obtained by uploading the *latest useful* chunk to the *most deprived* peer. The idea of sending data to the most deprived node in a network can be already found in [107], where random useful packets are forwarded.

The key component of this algorithm is the set of video chunks each peer P_i owns, which can be defined as $C(P_i)$. Assuming that the set of neighbours of P_i is N_i, the most deprived peer is the one that satisfies $\max_{P_k\in N_i} \{|C(P_i) \setminus C(P_k)|\}$. In other words, it is the peer that maximises the amount of chunks a certain peer owns and its neighbours do not. According to the original study [107], optimal streaming rates are obtained when random chunks are forwarded. However, the experimental evaluation performed in [106]

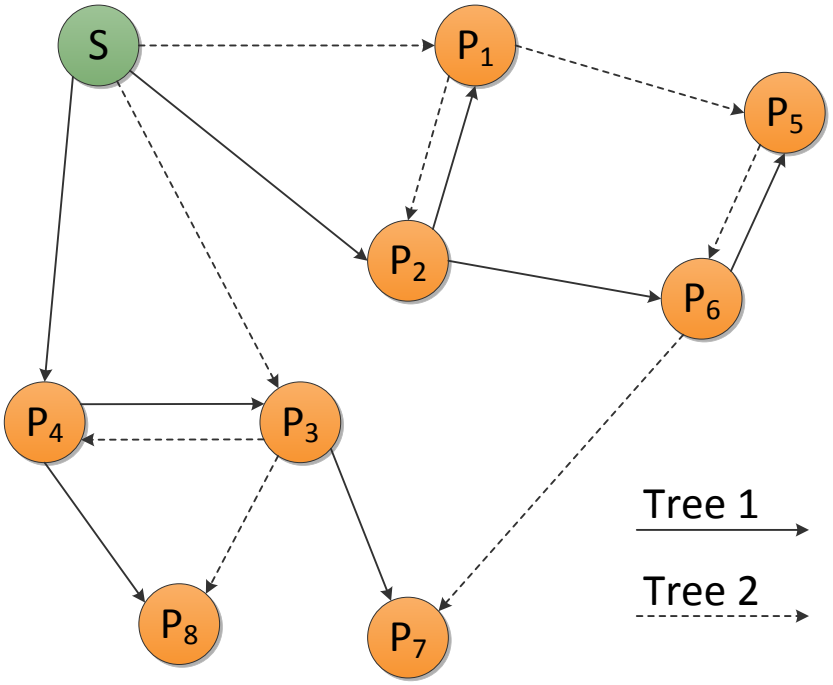


Figure 5.3: An example of SPPM overlay network. Two complementary multicast trees are maintained to route the data from the source of the stream to the set of users.

suggests that other techniques, for example forwarding the *latest useful* chunk, achieve similar results. An important remark is that the "latest useful" chunk in this context should be intended in terms of timestamp (e.g. position in the video bit-stream).

5.3.6 Stanford Peer-to-Peer Multicast

The system proposed in [108] is based on Stanford Peer-to-Peer Multicast (SPPM). This is a push-tree solution where several complementary multicast trees are created. These trees form the overlay network, whose root node is the source S of the video. Different packets are distributed using different trees. Figure 5.3 shows an example with two; in this case odd and even packets are distributed through the two possible different paths from S to peers $P_1 \dots P_8$.

The use of a push-tree solution as opposed to mesh-pull can lead to a better performance of the system [108]. However, creating and maintaining trees might be computationally expensive, especially in big networks. In this case, if a new peer wants to

join the network, it needs to contact the source. Similar to a BitTorrent tracker, the source returns a random list of peers. The new peer then selects the best parent nodes according to available throughput, distance from the source in logical hops and end-to-end delay. Finally, it connects to only one parent for each of the multicast trees. Every time a peer leaves the swarm, its children need to repeat this procedure and connect to another parent. Since connecting to a descendant creates loops that are not connected to the source, each peer keeps a list of all its ancestors (for each tree). While connected to the network, each peer will receive data from its parents and will forward it to its children.

As far as video scalability is concerned a "content-aware prioritisation" is used in case of network congestion. Each packet is assigned a different importance, which depends on the distortion-rate reduction given by its decoding. The packet's dependencies are also considered. That is, if a layered video is used, the base layer of the sequence has the highest priority, while enhancement layers are only transmitted if there is enough spare capacity in the system.

This tree-based solution does not consider potential free-riders. In fact, a peer might connect to several parents, while refusing to upload data to its children. In other words, a parent will always upload data to its children, if the network conditions allow it, regardless of their behaviour. The problem of peers tampering with the data is also not considered. However, this issue could be easily solved requesting a hash of each packet from a trusted entity (e.g. the source S).

5.3.7 Other Systems and Final Considerations

Finally, an experimental study [109] performed by Liang et al. addresses the issue of performance versus complexity of the system, proposing different scheduling techniques. This study is also important because it shows the impact of the network topology on the network and it suggests that peers with higher capacity should be linked to a larger number of peers. Social networking aspects are not considered here, though.

On the other hand, another study, performed specifically on PPLive [110] shows that the behaviour of IPTV users is the same as regular users. This can be a critical factor, as many peers might simultaneously leave at the beginning of commercials breaks and

List of Symbols

P_i	Peer i
c_i	Upload capacity of P_i
q_i	Quality layer i
b_i	Bit-rate of quality layer q_i
b_t^i	Target bit-rate for P_i
Δ_{max}	Maximum acceptable credit or debit a peer can have towards another peer
S	Content source
N	Set of peers in the network, including the content source
$\|N\|$	Number of peers in the network, including the source
N_i	Neighbours of P_i
N_i^*	Non free-riding neighbours of P_i
$\chi_k^s(i,t)$	Number of chunks P_k has sent to P_i at time t
$\chi_k^r(i,t)$	Number of complete and unpolluted chunks P_k has received from P_i at time t
$\delta(P_i)$	Deprivation of P_i
φ_{dep}^i	Deprivation factor of P_i, as computed by another peer
φ_{cr}^i	Credit factor of P_i, as computed by another peer
α, β	Weighting factors for φ_{dep}^i and φ_{cr}^i respectively
r_i	Chunk request, sent from P_i
$s(r_i)$	Score assigned to r_i

Table 5.2: List of symbols in use for the proposed approach.

re-join the network at the end of them. This article also points out that slow start is a problem that needs to be specifically addressed.

5.4 Proposed Approach

Our approach is based on the conclusions drawn from the simplified scenario. We are now considering peers with limited upload capacities and different upload speeds. A

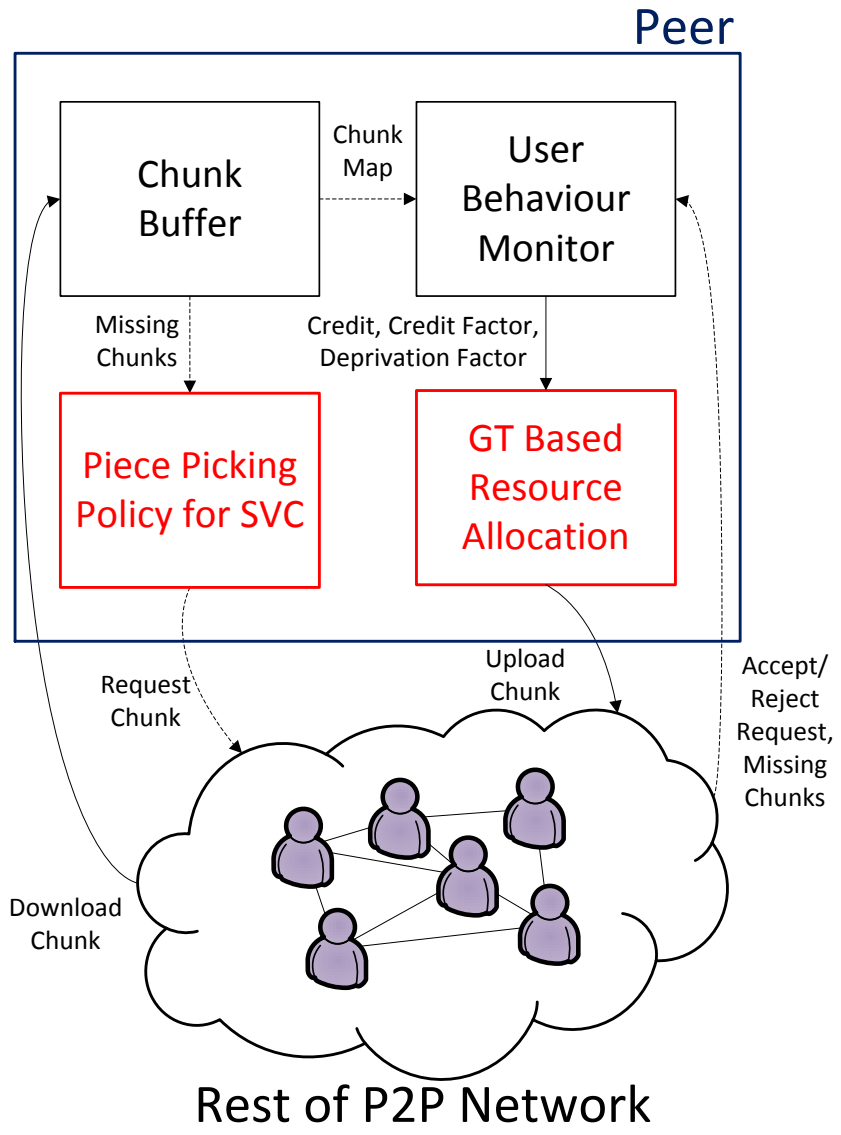


Figure 5.4: Block diagram for the proposed approach.

fraction of the user behaves as a free-rider. We are simulating the network using ns-2 [111] simulator, to deal with network unreliability. Video scalability helps achieve fairness among users. However, we still assume that peers cannot leave the network during the whole simulation and we also create the overlay network according to the standard BitTorrent protocol. That is, we are using a BitTorrent tracker.

This contribution is also published in [13, 16]. It is a credit-based framework for scalable P2P video streaming with a focus on optimal resource allocation. Our idea

is to integrate minimum-delay streaming functionalities within an incentive provision mechanism. Scalable video coding forms an integral part of our framework, as peers are discouraged from downloading videos with a quality which is higher than the one they can provide other users with. To the best of our knowledge, optimising streaming rates and finding incentives for users to cooperate are usually considered as separate problems. Most real systems, however, need to deal with both at the same time. Our technique considers both issues, while aiming at achieving nearly optimal performance. An important remark is that this approach is not compatible with the original BitTorrent. Therefore, for the purpose of testing we have not integrated this approach within a P2P client, but we used ns-2 network simulator, as it allowed to perform tests with a higher number of users.

The block diagram in Figure 5.4 summarises our proposed technique. The blocks representing our contributions are the Chunk Selection and Resource Allocation in the reference diagram (Figure 4.1). The first one is a modification of the technique proposed in Chapter 4, as described in the following section, while the second is a framework for resources allocation that takes decisions based on the behaviour of the peers (such as number of accepted requests) and information regarding their downloaded chunk map. In the figure, "Credit" indicates whether a peer has run out of credit and needs to be marked as a free-rider. Finally, credit and deprivation factors will be defined in Section 5.4.4. A list of the symbols used in the next sections can be found in Table 5.2.

5.4.1 Piece Picking Policy

This packet scheduling algorithm is similar to our solution presented in Section 4.4.1, however, due to completeness, we will briefly describe it, illustrating the key differences. A sliding window, containing a fixed number of GOPs that follow the current playback position is defined. The number of GOPs inside the window usually corresponds to 10-20 seconds of video. Inside the window, chunks have a priority that depends on the quality layer they belong to. Inside each layer, the rarest pieces are requested first. In contrast with the approach described in Section 4.4.1, however, this also holds for the base layer. In fact, as the playback positions of the peers are synchronised, a sequential policy would imply that many peers would be interested in the same chunk, whereas applying a rarest-first policy increases diversity. At regular intervals, the system performs a window shift

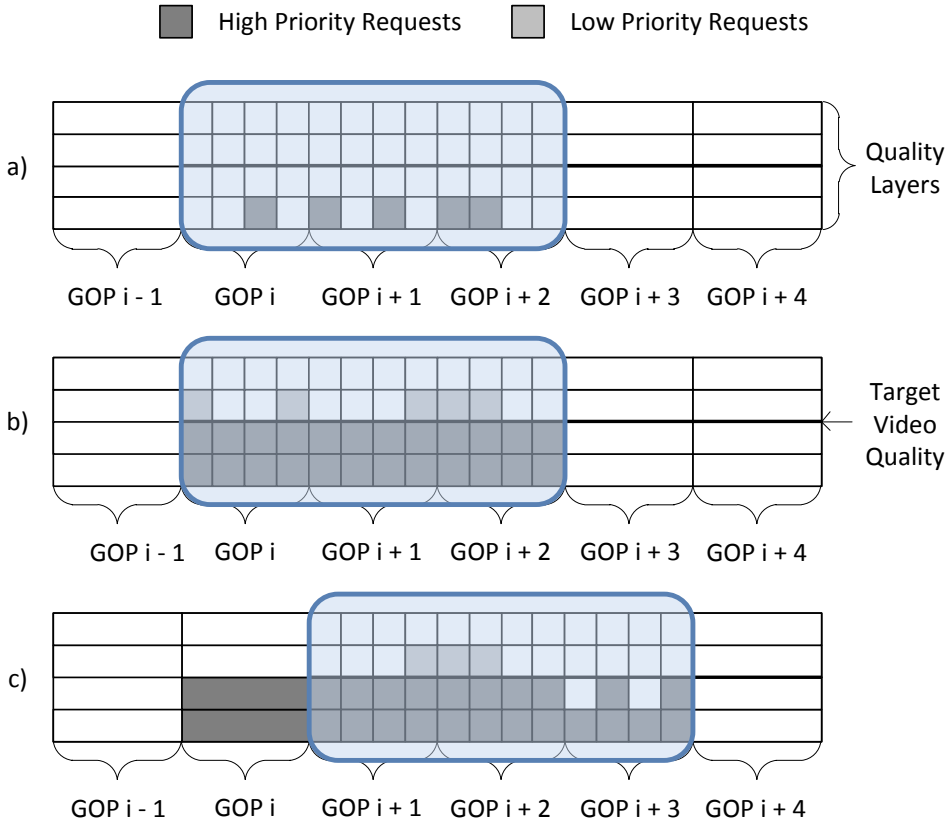


Figure 5.5: Sliding window for scalable live video streaming, showing high and low priority requests.

and checks how many quality layers of the current GOP have been completely received. If at least the base layer has been received, it will be decoded, otherwise the GOP will be skipped, as in this case the system cannot be paused. In this case, the received video bit-rate will be set to 0.

The sliding window is also shown in Figure 5.5. In a) the peer has just joined the network. It calculates the current playback position from its neighbours and there is a short pre-buffering before the first window shift. Figure 5.5 b) shows the final moments of the pre-buffering. The peer has already downloaded all the chunks it could afford to upload and it sends low-priority requests to its neighbours. After the window shifts in c), the completely received quality layers are decoded, while the partial ones are discarded.

5.4.2 Credit-Based Framework

In our system, a credit line mechanism is used. However, as it was previously introduced, two different types of requests are defined: regular – or high priority – and low priority. Regular requests, when accepted, will be considered when computing the calculation of the current credit, while the latter will not be taken into account. This is necessary, as we prove that peers should not *request data they cannot afford to upload*, but on the other hand we would also like to exploit the spare resources in the system. We consider the case in which peers have different upload capacities and we use SVC. Under these circumstances, *quality layers* are subsets of the original bit-stream associated to different SNR.

Proposition 5.1. *Let us consider a peer P_i whose upload capacity is c_i and a layered scalable video whose bit-rates of layers $q_0, \ldots, q_{max}$ are $b_0, \ldots, b_{max}$. If this peer downloads data belonging to quality layers $q_j, \ldots, q_{max}$, such that $b_j > c_i$, it will be eventually marked as a free-rider by its neighbours.*

Proof. A peer P_i is marked as a free-rider by its neighbours $P_k \in N_i$ if $\chi_k^s(i,t) - \chi_k^r(i,t) > \Delta_{max}$ for any $k \mid P_k \in N_i$, where N_i is again the set of neighbours of P_i, $\chi_k^s(i,t)$ is the number of chunks P_k sent to P_i at time t, $\chi_k^r(i,t)$ is the number of complete and unpolluted chunks P_k received from P_i at time t and Δ_{max} is the credit line set by all the peers in the network. Considering the most favourable case for peer P_i, which is when $\chi_k^r(i,t) = \chi_i^s(k,t)$ (e.g. there are no transmission errors), P_k will cooperate with P_i if $\lim_{t\to\infty}(\chi_i^s(k,t)/\chi_k^s(i,t)) = 1$. If this condition is not satisfied then, for an arbitrarily long time t', $\chi_k^s(i,t') - \chi_i^s(k,t') > \Delta_{max}$ will be verified. As χ^s depends on the upload capacity of a certain peer, a user should not try to download more data then it can afford to upload. Therefore, for the specific case of scalable video transmission, a peer should not aim to download a video with a bit-rate that is higher than its upload capacity. It also follows that free-riders and users with an upload capacity that is lower than the bit-rate of layer q_0 will be cut out of the system. □

This result can also be interpreted as follows: if a peer tries to download more data than it can upload, its behaviour is comparable to free-riding. In some cases, however, there might be some spare resources in the system. Therefore, in order to fully exploit the available capacity, low priority requests are defined. These are requests that concern

quality layers with a higher bit-rate than a peer can afford to upload. They will only be satisfied if the peer that receives one is not fully utilising its capacity and they will not be added to the current debt or credit.

It is intuitive to prove that it is not convenient for a peer to ask for all the chunks as low priority requests. For instance, no chunks belonging to the base layer can ever be requested under these circumstances. This would imply that the peer that sends the request does not meet the minimum requirements to be allowed inside the P2P network. Moreover, as far as enhancement layers are concerned, requests will only be satisfied if there is enough spare capacity in the system. In this case, a peer that can afford to upload a certain layer which asks for chunks belonging to the same layer with low priority only lowers the chance of having this request accepted and therefore has no interest from unilaterally changing its strategy.

5.4.3 Resource Optimisation

This algorithm aims to achieve real-time streaming through accurate resource allocation. In applications such as live streaming, it is fundamental to keep the upload channels of all the peers constantly busy. It has already been proven [105] that in order for the peers to be fully served, for a non-scalable video the following condition needs to be satisfied:

$$\sum_{i \mid P_i \in N} c_i \geq (|N| - 1) \times s, \tag{5.17}$$

where N is the set of peers (including the source), c_i is the upload capacity of a peer P_i, $|N|$ is the total number of peers and s is the video bit-rate. It is possible to extend the condition stated in Eq. (5.17) to scalable video sequences.

Lemma 5.1. *Assuming that a scalable video with bit-rates $b_1, \ldots, b_j, \ldots, b_{max}$ is used, and each peer P_i has a target bit-rate b_t^i, in order to achieve these received bit-rates the following condition needs to hold:*

$$\sum_{i \mid P_i \in N} c_i \geq \sum_{i \mid P_i \in N \setminus \{S\}} b_t^i, \tag{5.18}$$

where S is the source of the video.

Proof. Considering the total flow of data entering and leaving each peer, the Lemma follows. □

We now assume that the number of peers in the network is high and the contribution of the source is therefore negligible. Moreover, in order for a peer not to be marked as a free-rider, following from the proof of Proposition 5.1 the outgoing flow of data on every single channel should roughly correspond to the incoming one. In this specific case, the former condition can be rewritten as follows:

$$c_i \geq b_t^i, \qquad \forall i \mid P_i \in N \backslash \{S\}. \tag{5.19}$$

The two terms of the inequality shown in Eq. (5.19) can only be equal in case of optimal resource allocation, for example the one that fully exploits the upload capacity of all the peers. In some cases, however, this does not happen as some peers do not have any data their neighbours are interested in.

This problem can be partially overcome by uploading more data to the *most deprived* peers [106]. Considering the set of useful chunks a peer has (where useful could mean those inside a sliding window), its most deprived neighbour is the one who does not own the largest number of chunks in this set.

5.4.4 Optimal Policy for Resource Allocation and Free-riding Detection

As far as our resource allocation algorithm is concerned, a peer P_k adopts the following strategy when receiving a request r_i from P_i: First of all, every time a request is received, it is associated with a time-stamp and a time-to-live (TTL). Moreover, it is associated with a *score* $s(r_i)$. This score is defined as follows:

- If $\chi_k^s(i,t) - \chi_k^r(i,t) \geq \Delta_{max}$, $s(r_i) = -\infty$; this request will be immediately rejected, as P_i might be a free-rider.
- If $\chi_k^s(i,t) - \chi_k^r(i,t) < \Delta_{max}$ and the request has been sent with high priority, the request will be inserted into a buffer and it will be associated with the following

score:

$$s(r_i) = \left(\alpha \cdot \varphi^i_{dep} + \beta \cdot \varphi^i_{cr}\right), \tag{5.20}$$

where φ^i_{dep} is the *deprivation factor* and φ^i_{cr} is the *credit factor* of P_i, and α and β are arbitrary constants which satisfy the condition $\alpha + \beta = 1$. These quantities will be explained in detail in the following paragraphs.

- If $\chi^s_k(i,t) - \chi^r_k(i,t) < \Delta_{max}$ and the request has been sent with low priority, the request will be inserted into the buffer with $s(r_i) = 0$.

Every time a peer has finished sending a chunk, or if it is not sending any chunks, it will select a piece to upload. Among all the requests in its buffer, it will accept the one with the highest score, while the others will remain in the buffer until their TTL expires. The TTL for each request is not fixed and it depends on the current arrival rate of both high and low priority requests from its non-free-riding neighbours. If two or more requests have the highest score, the peer will forward the chunk that has been forwarded the least amount of times. An important remark is that low priority requests will be only accepted if there is enough spare capacity in the system.

As far as the quantities in Eq. (5.20) are concerned, the *deprivation factor* is calculated as follows. Considering all the non-free-riding neighbours N^*_k of P_k, for every peer P_j in this set the number of chunks P_k could contribute to them are counted. This value is also called the *deprivation* of peer P_j, or $\delta(P_j)$. The deprivation factor φ^i_{dep} of a peer is therefore computed as:

$$\varphi^i_{dep} = \frac{\delta(P_i)}{\sum_{j \in N^*_k} \delta(P_j)}. \tag{5.21}$$

In other words, φ^i_{dep} is the ratio between the number of chunks P_i is missing and the total number of chunks the non-free-riding neighbours of P_k are missing. It should be pointed out that the denominator in Eq. (5.21) cannot be zero when there are requests in the buffer. In fact, if $\delta(P_i)$ is zero, it means that there is no possible contribution towards P_i and the request makes no sense. Therefore, the denominator can only be null if there are no possible contributions towards any of the peers, which implies that the peer will not receive any requests and this calculation would not make sense. On

the other hand, the *credit factor* φ_{cr}^{i} indicates the contribution received from another peer and can be expressed as:

$$\varphi_{cr}^{i} = \frac{\chi_k^s(i,t) - \chi_k^r(i,t) + \Delta_{max}}{\sum_{j \in N_k} \chi_k^s(j,t) - \chi_k^r(j,t) + \Delta_{max}}. \tag{5.22}$$

It should be pointed out that $\chi_k^s(j,t) - \chi_k^r(j,t) + \Delta_{max}$ in Eq. (5.22) is zero only if a peers has run out of credit, in which case it does not make sense to compute this factor, as the request will be rejected in any case. Therefore, the denominator in Eq. (5.22) cannot be zero either. The deprivation factor helps achieve nearly optimal performance, while the credit factor is used to encourage resource reciprocation.

These quantities are also explained in Figure 5.6 and 5.7. In Figure 5.6, showing the credit factor, a peer gets a request rejected because it has already received too many chunks. For all the other cases, the peers are assigned a score that depends both on the contribution they gave and received. Similarly, in Figure 5.7 the sender peer computes the deprivation factors using the information from its own chunk buffer and its neighbours'.

5.4.5 Distribution of the First Copy

One of the critical aspects in P2P file sharing is the distribution of the first copy. Given a certain upload capacity of the source, the aim of the proposed algorithm is to maximise the data each peer has to share, increasing therefore the usage of their upload channels. In our approach, peers cannot request any data from the source, which only pushes data to selected peers in the swarm. First of all, the source identifies the most deprived non-free-riding peer in the swarm. This can be achieved using a G2G-like [9] algorithm. That is, peers will report to the source whenever they receive a chunk, specifying the user they received it from. Depending on this information, the source will first estimate if users have forwarded a number of chunks corresponding at least to the base layer of the sequence and depending on this they will classify users either as cooperative or deceptive. During the second phase, the source identifies the chunks this peer is missing. Among them, the seeder will then forward the one that has been forwarded the least amount of times.

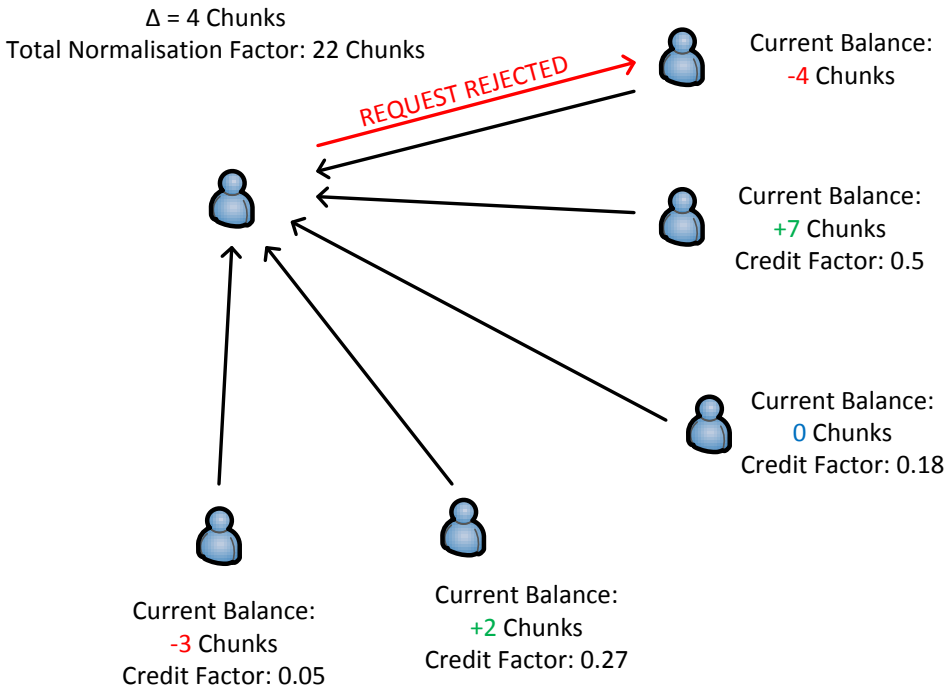


Figure 5.6: Example showing the credit factors of a peer's neighbours.

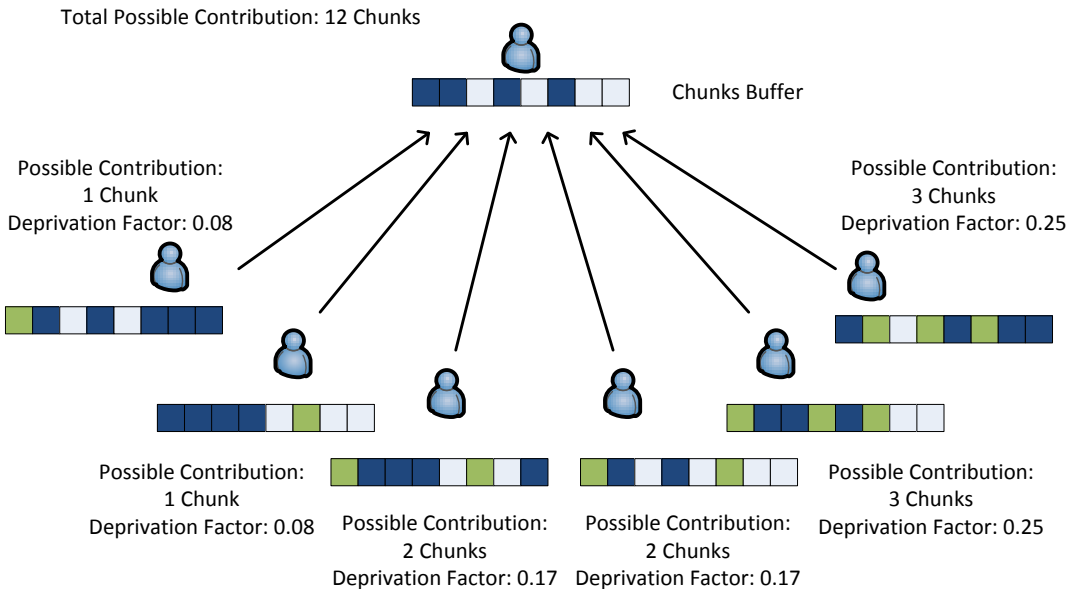


Figure 5.7: Example showing the deprivation factors of a peer's neighbours.

5.5 Implementation

For our experimental evaluation, we used a modified version of the ns-2 [111] network simulator. It is a "discrete event simulator targeted at networking research". It supports simulation of TCP. Moreover, each *Node* in the network can be associated with an *Agent* which generates packets and an *Application*, which performs several tasks (e.g. behave according to a certain protocol).

The starting point was the BitTorrent simulation in ns-2 described in [112]. Both packet-level and flow-level simulations are supported, however for our testing environment we only considered packet-level. A few modules have been added to provide an interface between the P2P network and WSVC sequences and implement the reputation mechanism described in the previous section. A new Application was created, which is based on *BitTorrentApp*, proposed in [112]. Implementation of the proposed techniques and integration with the rest of the simulator required over 2500 lines of code written in C++ [113] and additional 150 lines written in tcl [114] were needed to set all the simulation parameters, generate the nodes, the links and the routing table.

5.6 Results

For the purpose of simulation, City sequence – already used in previous evaluations – is repeated 15 times (for a total of 150 seconds) and encoded in WSVC format. The spatial resolution is 352×288 (CIF) and the frame rate is 30 fps. The video sequence is divided into chunks of 4 kB. In addition, there are 7 quality layers, with bit-rate r_b ranging from 256 kbps to 768 kbps. Information about WSVC GOPs and quality layers is stored in a metadata file, which we assume to be available to all peers. We decided to use only one sequence as WSVC allows to specify bit-rates manually and the differences between layer sizes are very small (no more than 30 bytes in most cases, much smaller than the defined BitTorrent chunk size). Table 5.3 shows a comparison among the cumulative number of chunks that are necessary to decode a quality layer for City, InToTree and Soccer when their extraction points are the same. As these values are the same and the actual content of these chunks does not have an impact on the performance of the

Video / r_b	City	InToTree	Soccer
256 kbps	17	17	17
320 kbps	21	21	21
384 kbps	26	26	26
480 kbps	32	32	32
576 kbps	38	38	38
672 kbps	44	44	44
768 kbps	51	51	51

Table 5.3: Cumulative number of chunks forming a quality layer for City, InToTree and Soccer sequences.

simulated system, we consider our results obtained with City to be also valid for slow (InToTree) and fast-moving sequences (Soccer).

In our experiments, only one source has the full sequence at the beginning. Peers are divided into two categories: regular users and free-riders. Regular users accept or reject requests according to the proposed algorithm, while free-riders reject any request they receive. We assume all the peers to join the system within a one-second interval, in order to keep their playback positions synchronised. After a pre-buffering phase, the playback starts.

In our tests, the network consists of 11, 31, 51 or 101 peers. This includes a video source, which has the same bandwidth as the other users. We consider scenarios with different upload capacities (50, 75 and 100 kB/s), fractions of free-riders (0%, 33%, 50% and 66%) and values of the credit line. Moreover, α and β range between 0 and 1, with $\beta = 1 - \alpha$. A comparison with an existing technique [85] is also shown. In the next paragraphs, the outcome of our experimental evaluation will be illustrated.

Figure 5.8 shows the impact of different credit lines on the behaviour of the system when there are 11 peers with no free-riders in the network and (α, β) is set to (0.5, 0.5). The credit line Δ_{max} grows exponentially with a power of 2 and it ranges between 1 and an infinite number of chunks. The average received video bit-rate grows as Δ_{max} grows up to a certain threshold, which in this case is 8 chunks. For values below the

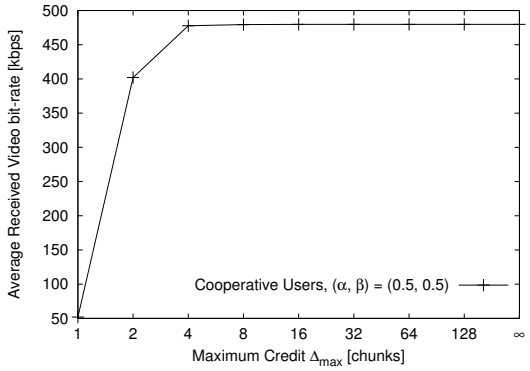
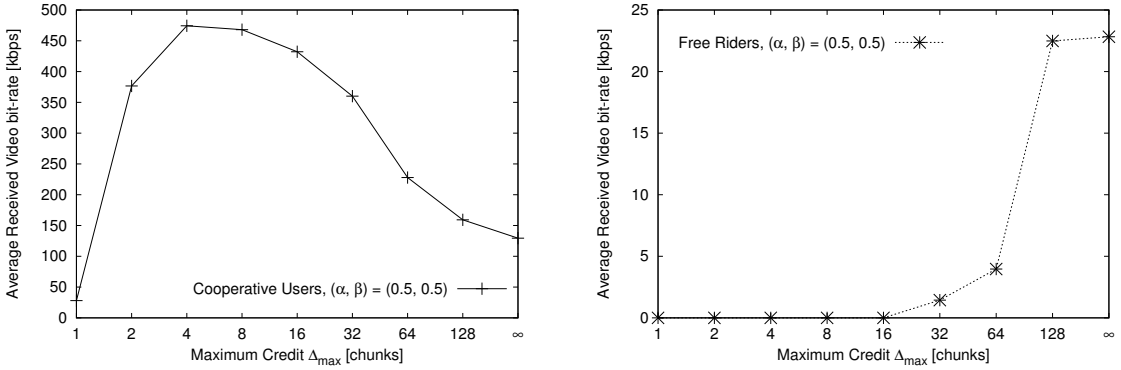


Figure 5.8: Average received video bit-rate with 11 peers, $c_i = 75$kB/s and 0% FRs.



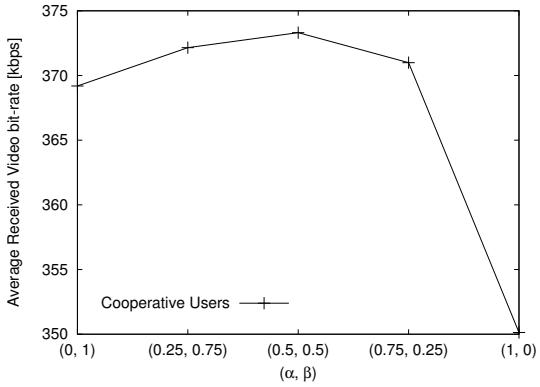
(a) Cooperative users.

(b) Free-riders.

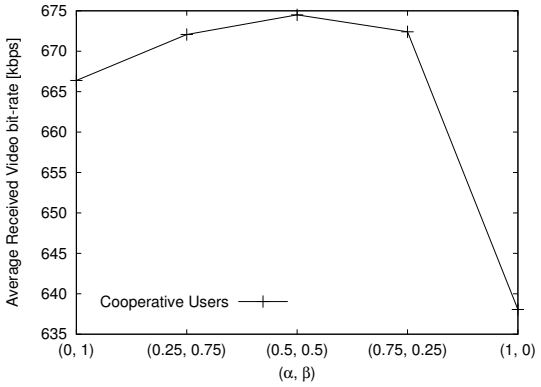
Figure 5.9: Average received video bit-rate by (a) Cooperative users and (b) Free-riders with 31 peers, $c_i = 75$kB/s and 66% FRs.

threshold, cooperative peers are incorrectly detected as free-riders, which causes poor system performance. For $\Delta_{max} = 1$, the average received video bit-rate (about 50 kbps) is lower than the bit-rate of the base layer (256 kbps), therefore real-time streaming is not achieved in this case. However, with Δ_{max} set to 4 chunks, the performance of the system is nearly optimal. Finally for values of Δ_{max} above the threshold, the system is fully utilising its resources and the received video bit-rate remains the same even with a maximum debit or credit set to infinite.

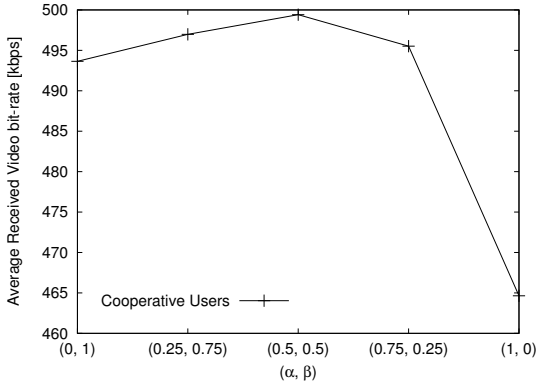
On the other hand, when 66% of the peers are free-riders as shown in Figure 5.9(a) and 5.9(b), using a Δ_{max} above 4 chunks causes a degradation in the received video bit-rate of cooperative users, while free-riders gain a benefit as the credit line grows. However, due to the G2G-like free-riding detection mechanism used by the source, the



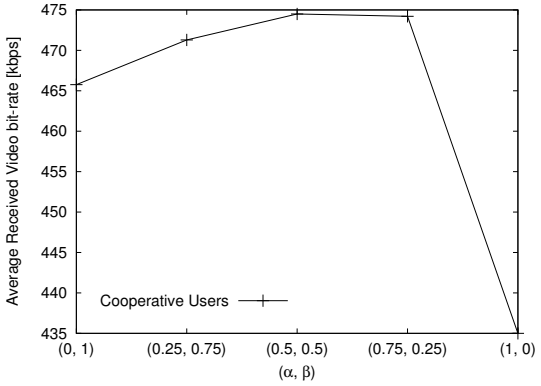
(a) Average received video bit-rate with 11 peers, $c_i = 50$kB/s and 33% FRs.



(b) Average received video bit-rate with 31 peers, $c_i = 100$kB/s and 33% FRs.



(c) Average received video bit-rate with 51 peers, $c_i = 75$kB/s and 50% FRs.



(d) Average received video bit-rate with 101 peers, $c_i = 75$kB/s and 50% FRs.

Figure 5.10: Average received video bit-rate by cooperative peers and FRs as a function of α and β varying the number of peers in the network, their capacity c_i, and the percentage of misbehaving users.

free-riders' credit factor being very low and the large amount of misbehaving peers in the network, their average received video bit-rate remains very low. Specifically, in this case they only manage to decode a very limited number of GOPs.

We are also considering the impact of parameters α and β on the behaviour of the system. The credit line is set to 4 chunks, as in most of the cases this value allows cooperative users to achieve the best performance, and there are 33% or 50% of free-riders in the network.

Before proceeding to the analysis of the results, let us now consider what would happen if a peer P_i changed the weights assigned to credit and deprivation factor unilaterally. First of all, by increasing α and uploading more data to deprived peers, P_i would not repay its debts towards another peer P_j as quickly as P_j's neighbours, increasing P_i's chance of being marked as a free-rider. On the other hand, by unilaterally increasing β, P_i would reciprocate resources more quickly, but not as quickly as P_i's neighbours would reciprocate theirs. Moreover, this would necessarily result in decreasing α. It is worth noticing that giving more importance to the deprivation factor presents a few similarities with optimistic unchoking in BitTorrent. This implies that by reducing α, P_i would decrease its chances of discovering good peers in the network, increasing the other peers' instead.

Figure 5.10 shows the results for (α, β) ranging from $(0, 1)$ to $(1, 0)$ for different numbers of peers. An important remark is that for $(\alpha, \beta) = (0, 1)$ the system is very similar to the one presented in [85]. These graphs indicate that when free-riders are in the swarm, considering both the credit and the deprivation factor gives better results than trying to solve the problem of resource reciprocation and delay minimisation alone. The graph in Figure 5.10(d), obtained with 101 peers, shows a very similar behaviour of the system with $(\alpha, \beta) = (0.5, 0.5)$ and $(0.75, 0.25)$. Despite the first set of values being the best, this result suggests that for bigger networks different values of (α, β) might lead to a better performance since, as the number of users increases so does deprivation, all the other parameters being equal. However, all graphs from Figure 5.10(a) to 5.10(d) indicate that $(\alpha, \beta) = (0.5, 0.5)$ gives the best performance. Therefore, our experimental evaluation verified that both factors are important, and more specifically equally important. That is, setting $\alpha = \beta$ results in a balance between accepting to forward data to the peer that has less to share – to increase the overall performance of the system, but also to potentially discover discover a good neighbour – and promptly repaying a peer's current debt, to avoid being marked as a free-rider. In fact, when increasing α over β (as a global parameter), the system is more vulnerable to free-riding, whereas in the opposite case, it is more vulnerable to starvation.

5.7 Conclusion

We described a framework for P2P scalable video transmission that exploits elements of credit-based systems and delay minimisation algorithms. We also defined credit and deprivation factors and analysed how giving them different weights had an impact on the performance of the system. The strongest aspect of our approach is that free-riders are cut out of the system, while cooperative users can achieve real time streaming. Moreover, using scalable video we also promoted fairness among users with different upload capacities. On the other hand, this system might be vulnerable to sudden changes in the network (churn) and handwash attacks, where free-riders leave the network and re-join it with a different identity. Finally, the free-riding detection mechanism described here only depends on the information collected by one peer. The effectiveness of this approach could be improved if the peers in the network could share information about the users they have dealt with. Therefore, in the next chapter, we will extend the work described here by adding social features to the system.

Chapter 6

Social-based Free-riding Detection

This chapter describes a social-based framework for P2P transmission. It is an extension of our research work presented in Chapter 5, which means the objective is again to identify free-riders and optimise resource utilisation, however this is now also achieved through exploitation of social relationships. In fact, peers that choose to become part of the social network are not anonymous entities anymore – as their identity can be verified – and interact either with other "social" users or regular peers. In our system, peers that have an identity and have established friendship relationships with other users can cooperate and share information about the rest of the network. The rest of the chapter is organised as follows: Section 6.1 describes the problem in detail; Section 6.2 presents the relevant background; Section 6.3 explains our proposed approach and how the social reputation of a peer can be calculated; Section 6.4 presents the results of our experimental evaluation; finally Section 6.5 concludes the chapter.

6.1 Problem Description

In our previous approach, each peer was trying to estimate what type of peer it was interacting with by analysing the neighbour's behaviour towards itself. However, the peer had no knowledge of other users' interactions with this peer. Supposing that this were common knowledge this could, in principle, help draw a more accurate picture of what type of users are in the network and clearly identify who the free-riders are. A reasonable approach would seem to gather all this information from all peers in the net-

work and build a centralised database where the past behaviour of each user is stored. This, however, presents two main challenges: first, handling all this information would incur in additional costs; and second – and more importantly – peers could lie and this information could not be considered fully reliable. It is however possible to build trust in P2P networks, and our solution for achieving this is to rely on social relationships, such as friendship. Therefore, the problem becomes understanding how to exploit these relationships for more accurate free-riders identification and punishment. More specifically, this task can be divided into:

- Identifying what type of information peers can share with other trusted users.
- Formulating an algorithm that uses this information to understand if another peer is a free-rider, building *de facto* a social reputation for that peer.
- Proposing a new tracker policy for building of the overlay network that allows users that trust each other to interact with the same unknown users.

6.2 Related Work

In this section we describe a few social-based P2P systems. Some of them introduce the concept of friendship among users, while others consider social features in terms of expected behaviour given the characteristic of a certain peer. All these systems have a common denominator: in contrast with early P2P systems, users are not anonymous entities that interact with other unknown peers; in fact, each of them is associated to an identity and its behaviour can in principle be traced. Other existing systems, not described here, propose a social network for data sharing based on P2P [115] or try to analyse trust and reputation systems in P2P [116].

6.2.1 Tribler BitTorrent Client: Social-based Features

BitTorrent client Tribler [68], already introduced in Section 4.3.3, can be considered an example of social P2P. In this system, social features have been introduced to tackle the following problems:

- **Decentralisation**: some P2P systems are not fully decentralised, for example they can have a single point of failure. Exploiting these features might help to achieve this goal efficiently, since most of the information could be shared only among people that belong to the same social group.
- **Availability**: adding social-based features to the system can incentivise peers to behave in a more altruistic way, for example they might share content for a longer period of time and consequently increase availability.
- **Integrity**. To reduce pollution in P2P systems, users can contact "trusted" peers, whose best interest is to actively participate in this process. These peers should avoid sharing corrupt or fake content, if they do not want their reputation to be ruined.
- **Incentives Provision**: social incentives can help reducing the problem of free-riding. In fact, users will tend not to "steal" resources from members of their own community and therefore they will be less likely to download from them without giving anything in exchange.
- **Network Transparency**: since the introduction of dynamic IP addresses, NAT boxes and firewalls, peers cannot send any content anywhere without the need of a trusted mediator, and again "trust" can be achieved using social-based features.

In Tribler, peers are de-anonymised and a log of all their previous activity is kept. This information is used to build a reputation of the peers, as well as a profile of a user's favourite content. Moreover, groups can be created and peers can get recommendations about a certain content (and download the corresponding torrent file) from users with similar preferences. Peers can also discover other users and files using epidemic protocols [68]. In addition to this, friendship is defined. If a peer is a friend of another peer, it can help it with the download of a file. The two peers will download two different sets of chunks and the "helper" will upload the data it received to its friend, without expecting anything in exchange. In the context of scalable video transmission, this approach can in principle help receive a better video quality, however, it requires that part of the upload bandwidth of the helper is dedicated to uploading data to its friend, and not to reciprocating resources of the peers it is downloading from. In P2P systems that use free-riding detection mechanisms, this may cause this peer to be marked as malicious.

6.2.2 P2P Protocols Based on Social Norms

This approach [117] is based on the concept that each peer in the network has a global reputation, which determines its strategy. A social norm is in fact made of two elements: first a social strategy, which "represents the rule prescribing to the peers when they should or should not provide contents to other peers based on their own reputation as well as the reputation of the requesting peers" and second a reputation scheme, which "rewards or punishes peers depending on whether they comply or not with the social strategy".

For each peer, a set of allowed actions $\mathcal{A} = \{S, NS\}$ is defined, where S corresponds to "Request Served" and NS to "Request Not Served". Moreover, peers also have a tag θ representing their global reputation or social status. θ is assumed to be a natural number belonging to $\Theta = \{0, 1, \cdots, L\}$; L is is the maximum reputation and it is a parameter of the system, while higher θ corresponds to higher reputation and reflects a cooperative behaviour in the past.

In this framework, peers will behave according to the following strategy σ:

$$\sigma(\theta, \tilde{\theta}) = \begin{cases} S & \text{if } \theta \geq h(\sigma) \text{ and } \tilde{\theta} \geq h(\sigma) \\ NS & \text{otherwise} \end{cases}. \tag{6.1}$$

Eq. (6.1) shows that there is a reputation threshold $h(\sigma)$ under which a θ-peer will not cooperate with a $\tilde{\theta}$-peer and will not receive a contribution, as this is what is requested by the social norm. Similarly, two peers with good reputation (above the threshold) are required to cooperate. If the peers follow what is prescribed by the social norm, their social reputation will increase, otherwise they will receive a punishment. The paper continues analysing different punishment schemes. For example, the reputation of a peer can be reset to 0 if it refuses to cooperate, or it can just be lowered. Punishments normally result in peers being excluded from chunk exchanges for a certain number of turns (until their reputation becomes high enough again). Effects of free-riders, peers that are always altruistic and transmission errors are also taken into account.

This paper [117] is important for two reasons: first, the attitude of a peer towards another user does not depend on its previous interactions with it, on the contrary it relies on its global social reputation. This implies that resource reciprocation is not

List of Symbols

P_i	Peer i
$R^i_{soc}(P_j)$	Social reputation of P_j according to P_i
F^i_j	Set of friends of P_i that have interacted with P_j
$\|F^i_j\|$	Number of friends of P_i that have interacted with P_j
Δ_{max}	Maximum acceptable credit or debit a peer can have towards another peer
Δ_{soc}	Social reputation threshold; can also be seen as "social credit"
$\chi^s_k(i,t)$	Number of chunks P_k has sent to P_i at time t
$\chi^r_k(i,t)$	Number of complete and unpolluted chunks P_k has received from P_i at time t
φ^i_{dep}	Deprivation factor of P_i, as computed by another peer
φ^i_{cr}	Credit factor of P_i, as computed by another peer
α, β	Weighting factors for φ^i_{dep} and φ^i_{cr} respectively
r_i	Chunk request, sent from P_i
$s(r_i)$	Score assigned to r_i

Table 6.1: List of symbols in use for the proposed approach.

necessarily direct anymore. Second, peers have different strategies and are expected to maintain a different behaviour according to their type. It is worth mentioning that the research work in [117] does not consider scalable video. Some ideas on how to exploit scalability in a similar scenario will be discussed in Section 7.2 as a starting point for future work. A critical aspect of this approach, however, is that it is based on a global reputation, which may be difficult to manage if the size of the network grows, and it might be vulnerable to collusion attacks, where malicious peers report wrong information to the reputation manager.

6.3 Proposed Approach

This proposed approach has been published in [17]. Its block diagram is shown in Figure 6.1. The chunk selection algorithm is the same as the one used in Chapter 5 and the main difference with the diagram in Figure 5.4 is the presence of a "Social Network"

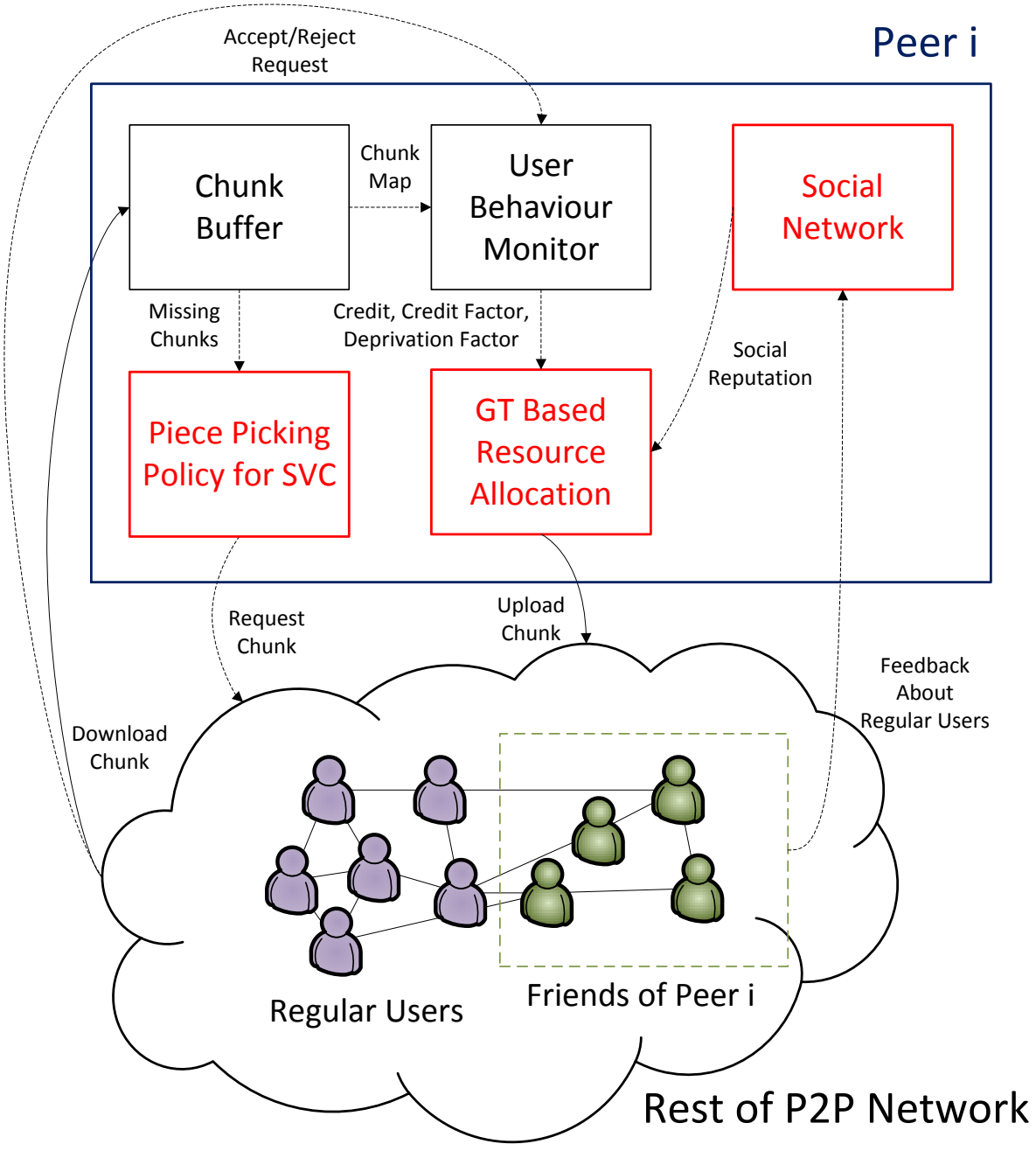


Figure 6.1: Block diagram for the proposed approach.

block, which elaborates the information received from friend peers and builds a social reputation for those peers which friends have interacted with. A list of symbols used in the description of this approach can be found in Table 6.1.

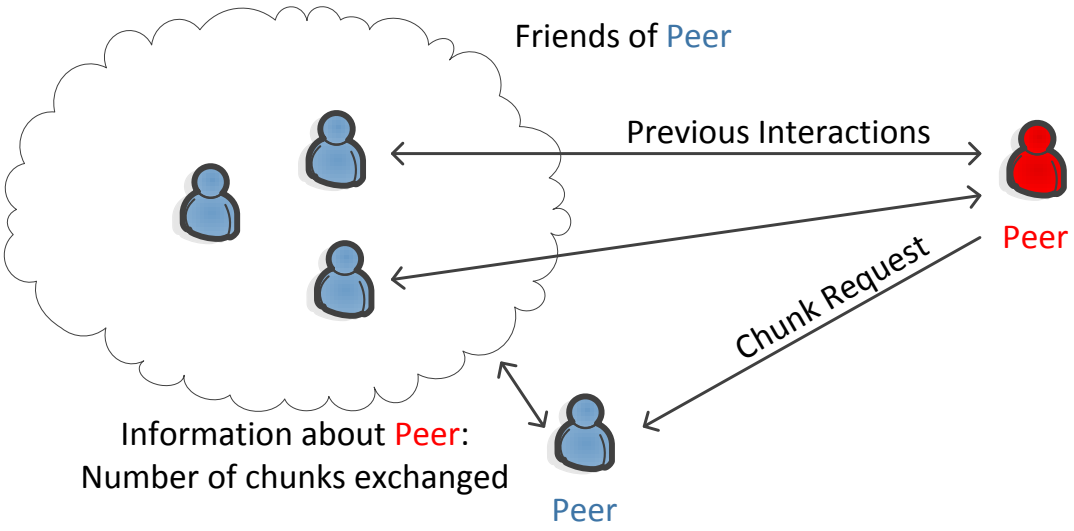


Figure 6.2: Social Reputation

6.3.1 Credit-Based Framework

Due to completeness, we briefly describe here the main characteristics of our credit-based framework. More details can be found in Section 5.4. A credit line mechanism is used and there is a maximum number of chunks a peer can download from another user without giving anything in exchange. If a peer requests more, it will be marked as malicious. However, as the experimental evaluation will show in Section 6.4, in the proposed framework we are relaxing the constraints on the maximum debit, giving less importance to direct reciprocation, as in practice we also want to consider indirect cooperation. Moreover, similarly to our previous approach, two different types of requests are defined: regular and low priority. This is necessary, as we proved with Proposition 5.1 in Section 5.4.2 that peers should not *request more data than they can afford to upload*, but we would also like to exploit the spare resources in the system. Low-priority requests will only be accepted if the peer that receives one is not fully utilising its capacity and they will not be considered for the credit or debit computation.

6.3.2 Social Networking Extension

The aim of this algorithm is to exploit information from trusted peers in the network (friends) in order to better identify malicious users. We start by defining three types of peers:

1. **Peers inside the social network**: We assume that they are always cooperative. In fact, not being anonymous, their behaviour can be easily tracked [9]. When choosing which peers to upload chunks to, they follow the strategy described in this chapter.

2. **Cooperative users outside the social network**: These users follow the strategy described in Chapter 5.

3. **Free riders**.

As far as our assumption regarding peers inside the social network is concerned, it is necessary to illustrate further considerations regarding potential social attacks. In Section 1.1 we already introduced Sybil attacks [10], where an *entity* presents itself to other entities in the system using multiple *identities*. In our system, a malicious user could potentially pretend to be several peers at the same time, requesting to establish friendship connections with users it does not actually know. In a recent study by Manago et al. [118] considering Facebook friendships established by a few students in an American University, 90% of the connections were with close friends, activity partners, previous connections or acquaintances, whereas 4% had been established with complete strangers. We believe that this scenario might be realistic also for P2P systems, or in a practical implementation of this architecture P2P users could be linked to existing social network profiles, and therefore the impact of Sybil users may be limited. However, even supposing that the percentage of friendship connections with unknown peers is higher than the one found in the study, and that a significant share of these are malicious or Sybil users, there are additional ways to tackle this problem:

i. as we already stated, since peers are not anonymous, their misbehaviour is easy to manually detect by cooperative users, which can remove the link;

ii. there exists an automatic mechanism that can detect Sybil nodes with a high interval of confidence, called SybilGuard [119].

SybilGuard considers a social network where regular (honest) and Sybil nodes are both present. When two nodes are connected by an edge, they are friends and if the edge is between an honest and a Sybil user, it is called an *attack edge*. The number of attack edges is limited by the number of connections the identities of a malicious entity can establish with the rest of the network. The protocol then considers random walks, which are defined in such a way that they verify a few properties when choosing an honest node as starting point and are not likely to when starting from a Sybil node, due to the limited number of attack nodes cooperative users can establish with the rest of the network as compared to honest users. More specifically, this algorithm is able to identify that several converging random walks start from Sybil nodes.

Regarding our approach, the credit line mechanism in Chapter 5 was introduced to add some tolerance when a peer could not reciprocate resources immediately. However, this mechanism only considered the information a single peer had about the rest of the network. In this context, on the other hand, when a peer needs to mark a user which is not a friend as cooperative or free-rider, it can ask its friends – which have interacted with this user – for the number of chunks they have exchanged and classify the peer according to this information. In a practical implementation, this information exchange could either be direct, or handled by the tracker. An illustration of the concept of *social reputation* is given in Figure 6.2. More specifically, the social reputation of peer P_j, according to P_i is defined as:

$$R^i_{soc}(P_j) \hat{=} \sum_{k|P_k \in F^i_j} \chi^s_k(j,t) - \chi^r_k(j,t), \tag{6.2}$$

where again $\chi^r_k(i,t)$ is the number of complete and unpolluted chunks P_k received from P_i at time t, $\chi^s_k(i,t)$ is the number of chunks P_k sent to P_i at time t and F^i_j are those friends of P_i which interacted with P_j. It is calculated as the overall credit or debt towards a peer's set of friends. This quantity needs to be compared to a threshold which is based on the number of friends an unknown peer has interacted with. Its definition is based on the idea that the larger the number of interactions is, the bigger the overall contribution to these peers should be. Therefore, the incremental allowance (in chunks) given by the presence of one more peer in F^i_j should decrease as a function of its size.

More formally, the social reputation threshold is defined as:

$$\Delta_{soc} \hat{=} \left\lfloor \Delta_{max} \sum_{n=1}^{|F_j^i|} \frac{1}{n} \right\rfloor , \tag{6.3}$$

where $|F_j^i|$ is the number of friends of P_i which have interacted with P_j. Therefore, if $R_{soc}^i(P_j) < \Delta_{soc}$, P_j is marked as cooperative, otherwise it will have a *bad social reputation* and P_i will refuse to cooperate with it.

We remind that that the optimal value for Δ_{max} in the non-social case, found in the experimental evaluation in Section 5.6 was a trade-off between minimising the number of cooperative users erroneously detected as misbehaving and minimising the amount of data uploaded to free-riders. By exploiting social features, it is possible to increase the value of Δ_{max}, as it will be shown in Section 6.4. This happens as free-riders will have a low social reputation, while cooperative users will have the chance to reciprocate resources to other peers.

6.3.3 Optimal Policy for Resource Allocation and Social Free-riding Detection

As far as our social-based resource allocation algorithm is concerned, a peer P_k adopts the following strategy when receiving a request r_i from a peer P_i: every time a request is received, it is again associated with a time-stamp and a TTL. Moreover, it is associated with a score $s(r_i)$, defined as follows:

- If $\chi_k^s(i,t) - \chi_k^r(i,t) \geq \Delta_{max}$, or the peer has a bad social reputation, $s(r_i) = -\infty$; this request will be immediately rejected, as P_i might be a free-rider.
- If $\chi_k^s(i,t) - \chi_k^r(i,t) < \Delta_{max}$ and the request has been sent with high priority, the request will be inserted into a buffer and associated with a score $s(r_i) = \left(\alpha \cdot \varphi_{dep}^i + \beta \cdot \varphi_{cr}^i\right)$, where again φ_{dep}^i and φ_{cr}^i are the the deprivation and credit factors of P_i – defined in Section 5.4.4 – as computed by P_k and α and β are arbitrary constants that satisfy $\alpha + \beta = 1$.

- If $\chi_k^s(i,t) - \chi_k^r(i,t) < \Delta_{max}$ and the request has been sent with low priority, the request will be inserted into the buffer with $s(r_i) = 0$.

It is important to notice that the main difference with respect to the system in Chapter 5 is represented by adding social reputation to the first condition. Similarly to our previous approach, every time a peer sends a chunk, it will accept the request in the buffer with the highest score, while the others will remain in the buffer until their TTL expires. The TTL for each request depends on the current arrival rate of both high and low priority requests from its non-free-riding neighbours. If two or more requests have the highest score, the peer will forward the least forwarded chunk.

A short proof of concept in favour of our social-based approach is presented in Lemma 6.1.

Lemma 6.1. *Let us suppose that a free-rider P_{FR} is interacting with N peers. Assuming that none of these are friends, the free-rider can download $(N\Delta_{max})$ chunks before being marked as a free-rider by all of them. If all N peers are friends, the maximum number of downloaded chunks is $\Delta_{max} + \lfloor \Delta_{max} \sum_{n=1}^{N-1} \frac{1}{n} \rfloor - 1 \leq (N\Delta_{max})$ for $N \geq 2$.*

Proof. When no social relations are considered, the total threshold $(N\Delta_{max})$ is just the sum of all the credit granted by all the peers. On the other hand, considering a peer P_i and its friends, the free-rider has a maximum allowance Δ_{max} granted by P_i and a "social" allowance or credit $\lfloor \Delta_{max} \sum_{n=1}^{N-1} \frac{1}{n} \rfloor$, according to Section 5.4.4 and Eq. (6.3). As we already stated, a free-rider will be identified by P_i if it either runs out of its non-social or social credit. Therefore, if it downloads $\Delta_{max} - 1$ chunks from P_i and $\Delta_{soc} - 1$ from P_i's neighbours (the latter corresponds to the social reputation $R_{soc}^i(P_{FR})$), the free-rider cannot have been identified yet at least by P_i. It is important to note that since the free-rider is interacting with all the peers in this set, Δ_{soc} is the same for all these users.

As soon as the free-rider downloads a chunk from any peer in the set, it is marked as a free-rider by P_i, but nothing can be said yet about any of the other peers P_k. If the free-rider has downloaded $\Delta_{max} - 1$ chunks from a user P_k', the same argument holds for this peer. On the other hand, if the free-rider has downloaded less than this value from any other P_k'', it must have downloaded more chunks from the neighbours of P_k'' (since the total number of downloaded chunks stays the same) and $R_{soc}^{k''}(P_{FR}) > R_{soc}^i(P_{FR})$. Therefore, since we assumed that $R_{soc}^i(P_{FR}) = \Delta_{soc} - 1$, the free-rider must have a bad

social reputation according to P_k''. The maximum number of chunks the free-rider can download is thus the one stated by the Lemma. □

On the other hand, under the same circumstances cooperative users not in the social network will have bigger chances to reciprocate their resources to any of these peers and will therefore not be affected by this detection mechanism.

6.3.4 Tracker

P2P networks can in principle consist of several hundreds of users. If peers are connected to random users, the chance that a friend is connected to the same peer is very low. Therefore, if friend peers are connected to different users it is impossible to build a social reputation. This problem can be overcome while building the overlay network.

The behaviour of the tracker is the following:

- If a **peer inside the social network** connects to the network, requesting a list of users, the tracker returns a list of its friends currently connected to the network.
- If a **peer outside the social network** (cooperative or free-rider) connects to the network, the tracker returns a list of peers in the social network and their friends.

By doing so, we make sure that peers outside the social network are always connected to users that can exchange information about them.

6.4 Results

This technique, similarly to our previous approach, has been implemented using ns-2 [111] simulator. It required an additional 500 lines of code with respect to the system described in Chapter 5 to manage the social network and the social reputation. In addition, we generated a graph which satisfied the Watts-Strogatz small world topology requirement [42] using the graph generator of Pajek [45]. As far as the parameters mentioned in Section 2.4 are concerned, the number of users N was 50, $K = 4$ and $p = 0.55$. They satisfy the requirements of Eq. (2.1) with $K > \ln(50) = 3.91$ and the resulting graph satisfied the property of connection. Therefore, it became the social network used

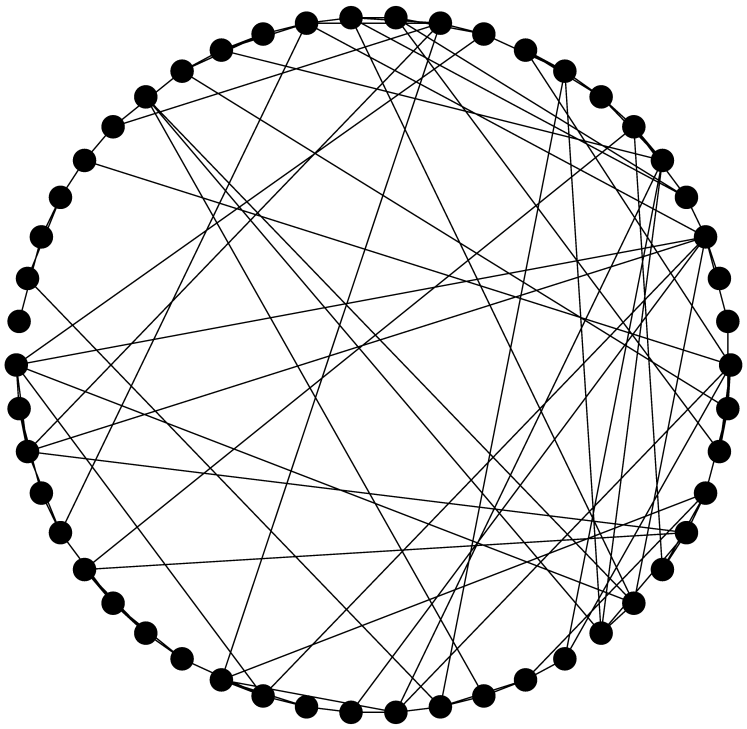


Figure 6.3: Social Network used in our experimental evaluation.

for our experiments, which is also shown in Figure 6.3[1]. In Section 2.4 we already stated the limitations of this model, however, our experiments are performed on a relatively small network, and we are interested in analysing the impact of information coming from trusted peers without considering "friends of friends" relationships, therefore we used this simple algorithm. When performing experiments on a larger scale, other models like the "forest fire" [47], also explained in Section 2.4, should be used.

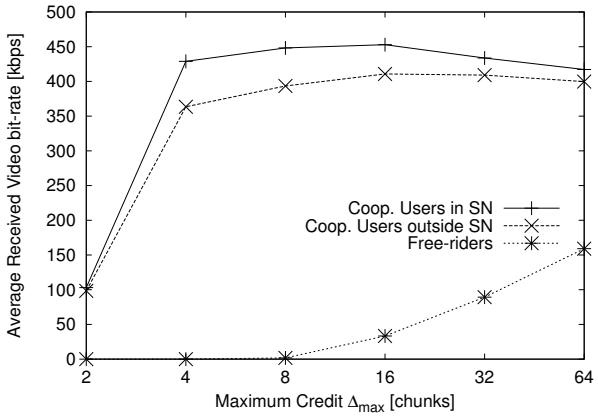
As far as the test sequence is concerned, City was repeated 15 times (for a total of 150 seconds) and encoded in WSVC format. The spatial resolution was 352×288 (CIF) and the frame rate 30 fps. The video sequence was split into chunks of 4 kB. In addition, there were 7 quality layers, ranging from 256 kbps to 768 kbps. In our tests only one source had the sequence and peers were divided into the three specified categories. We always had 50 peers in the social network, a variable percentage of free-riders and the

[1]This figure has been generated with Pajek [45].

remaining peers were cooperative users outside the social network. The peers joined the system within a one-second interval. After the pre-buffering, the playback started. Cooperative users had an upload bandwidth $c_i = 75$ or 100 kB/s. The size of the network in our sets of experiments was 81 or 121 peers. This included a video source, which distributed the first copy.

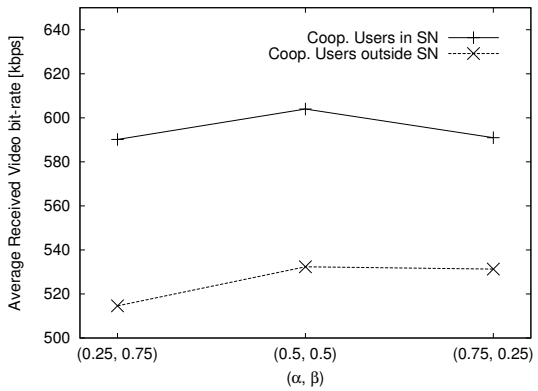
Figure 6.4 shows the behaviour of the free-riding detection mechanism as Δ_{max} is changing. There are 81 peers in the network, of which 20% free-riders. The upload bandwidth of the regular peers is 75 kB/s and the source's capacity is 150 kB/s. The graph shows that, similarly to our previous results in Section 5.6, the average received video bit-rate is low for very small values of the maximum credit due to false positives, as well as for a very large Δ_{max}, where on the other hand free-riders have more freedom of action. The average received video bit-rate presents a maximum for $\Delta_{max} = 16$. This value is bigger than the one found in our previous results. Our finding is that, as free-riders are also detected by analysing their social reputation, more tolerance can be added with respect to the reciprocation towards a single user, for which we also have already provided a short proof of concept in Lemma 6.1. An important remark is that users in the social network perform better than the others, as due to the social-based mechanism they upload less data to free-riders and more to users which reciprocate.

Figure 6.5 and 6.6 show the impact of the weights assigned to the credit and deprivation factor α and β on the behaviour of the system. Δ_{max} is set to 16 chunks and there are 81 and 121 peers. Free-riders are 20% and 33% of the total users and the upload bandwidth is 100 kB/s and 75 kB/s respectively. The source has now an upload capacity which is 150% of the cooperative peers'. The graphs show the average received video bit-rate for cooperative peers when (α, β) varies between $(0.25, 0.75)$ and $(0.75, 0.25)$. These graphs indicate that peers in the social network benefit from a higher received video bit-rate, and statistical analysis indicates that this difference is significant, as the t-test for $(\alpha, \beta) = (0.5, 0.5)$ gives $t > 10$ with $p < 2 \cdot 10^{-15}$ in both cases. Moreover, as far as the peers in the social network are concerned, the best weights for the deprivation and credit factor are $(\alpha, \beta) = (0.5, 0.5)$, which once again represent the best trade-off between trying to repay a peer's debt promptly and trying to provide peers with data which they can share. This result is also confirmed by the statistical analysis of variance (ANOVA), as $F > 7$ and $p < 0.05$ for both sets of experiments.



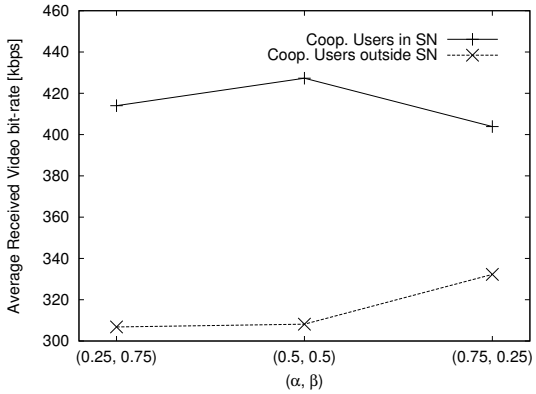
ANOVA; peers in SN for different Δ_{max}, $\Delta_{max} \geq 4$		t-test; peers in and out of SN for $\Delta_{max} = 16$	
$F = 14.14$	$p < 3 \cdot 10^{-10}$	$t = 6.83$	$p < 5 \cdot 10^{-9}$

Figure 6.4: Average received video bit-rate with 81 peers as a function of Δ_{max}, with c_i = 75kB/s, $(\alpha, \beta) = (0.5, 0.5)$ and 20% FRs.



ANOVA; peers in SN for different (α, β)	
$F = 3.08$	$p < 0.05$
t-test; peers in and out of SN for $(\alpha, \beta) = (0.5, 0.5)$	
$t = 10.5$	$p < 2 \cdot 10^{-15}$

Figure 6.5: Average received video bit-rate with 81 peers as a function of (α, β), c_i = 100kB/s, Δ_{max} = 16 and 20% FRs.



ANOVA; peers in SN for different (α, β)	
$F = 12.2$	$p < 1.5 \cdot 10^{-5}$
t-test; peers in and out of SN for $(\alpha, \beta) = (0.5, 0.5)$	
$t = 22.1$	$p < 10^{-15}$

Figure 6.6: Average received video bit-rate with 121 peers as a function of (α, β), c_i = 75kB/s, $\Delta_{max} = 16$ and 33% FRs.

An important remark is that Figure 6.6 indicates that $(\alpha, \beta) = (0.5, 0.5)$ is not the best value for the peers outside the social network. On the contrary, the graph suggests that the average received bit-rate increases as the deprivation factor increases. The differences with the other sets of experiments are: a higher number of peers, a lower bandwidth and a higher percentage of free-riders and the graph suggests that peers outside the social network are more vulnerable to these factors, which determines a worse performance. Our interpretation to the different impact of α and β is that these peers' worse performance implies that they have less chunks to share with the rest of the network. This results in a contribution to the other peers that is not enough to have a good credit factor. Therefore, giving more weight to the deprivation factor could improve their overall performance.

6.5 Conclusion

We proposed a framework for P2P scalable video transmission that exploited elements from social networking, credit-based systems and optimal network resource allocation algorithms. Peers that established social relationships with their neighbours could share information about the rest of the network and better identify free-riders. Experimental evaluation showed that, with respect to the approach described in Chapter 5, it was possible to increase the value of the maximum allowed credit and still be able to identify free-riders, thanks to our social-based detection algorithm. The results also confirmed that giving equal weight to the credit and deprivation factor still represented the best trade-off and allowed to maximise the received bit-rate. Finally, peers in the social network were rewarded by a better received video quality.

As far as the potential critical aspects of this approach are concerned, we identified the fact that peers always distinguish between free-riders and cooperative peers, not keeping the heterogeneity of the network into account. This suggests some ideas for future work, as peers for example could refuse to share their chunks belonging to upper quality enhancement layers to peers which are not considered "good" enough.

In the next chapter, we will summarise all the achievements described in this thesis and we will discuss how this work can be extended in the future.

Chapter 7

Conclusion and Future Work

This Chapter summarises the main contributions that can be found in this Ph.D. thesis in the field of scalable video transmission over P2P, as well as providing directions for future work.

7.1 Contributions

Our contributions described in this thesis can be summarised as:

- We performed a subjective video evaluation simulating different networking conditions using two different scalable codecs: WSVC and H.264/SVC. We defined realistic video transmission scenarios and asked test subjects to rank their visual experience under these different circumstances. These results were compared with an objective evaluation and the two codec themselves were compared both in terms of objective and subjective performance. The conclusion of this study was taken into account in the design of an actual P2P system.
- We designed a sliding window-based piece picking policy that requests the quality layers of the original scalable video bit-stream in a hierarchical fashion, so that the received video bit-rate can adapt to the current download bandwidth of a peer.
- We proposed a neighbour selection policy that only requests P2P chunks belonging to the base layer to peers that have proven to be good enough. This policy ranks all

the peers a user is currently downloading from according to their download speed and subsequently selects the best users in the network.

- We built a credit-based framework for optimal resource allocation in P2P whose joint aim is to cut free-riders out of the system. This technique uses a credit line mechanism, which means that there is a maximum difference in the number of video chunks two peers can exchange before the one which is in debt is marked as a free-rider. We defined two types of requests: high and low priority. They have a different impact on a user's reputation to exploit any idle resources in the network. For all the supposed non-free-riding peers, their requests are assigned a score which depends on two factors: their past generosity towards the peer that receives the request (credit factor) and the potential contribution they can receive from it (deprivation factor). We also showed how giving different importance to these two factors has an impact on the behaviour of the proposed system.

- We developed an algorithm that exploits the knowledge of a peer's friends to compute a third peer's social reputation and uses it to determine if that peer is a free-rider. It tries to estimate the behaviour of this unknown peer from the number of friends it has interacted with and the number of video chunks that have been exchanged with them. More specifically, this approach uses a credit line mechanism applied to a larger set of peers. The maximum total difference allowed depends on the size of this set, however, for each new interaction the additional granted credit decreases.

- Finally, for the social-based approach, we designed a tracker protocol that generates connections among users taking social relationships into account. Friend users are connected among each other and peers outside the social network are linked to groups of friend users. This way, free-riding peers can be detected by our social based algorithm.

7.2 Future Work

The new High Efficiency Video Coding (HEVC) standard [120], also known as H.265 [121] has been recently proposed. Its scalable extension is yet to be ratified, despite a few

preliminary techniques having already been suggested [122, 123]. We believe that the most sensible approach is first of all to research how our proposed algorithms can be adapted to the new standard. Therefore, we are closely following all the updates related to the development of this new codec. On the other hand, as far as the scalable codecs currently available are concerned, we will perform tests on longer sequences, also considering the impact of playback pauses, and we will not only focus on subjective video perception but also on QoE in terms of user engagement.

Moreover, possible directions regarding further development of our research work include further exploitation of social-based features in P2P systems. Some ideas include:

- The overlay P2P network should only correspond to a social network. In this case, peers are completely de-anonymised and it is very easy to detect free-riders and malicious users in general. However, this approach requires a peer's friends to be interested in the same content. Intuitively, if there are not enough friends currently in the swarm, the streaming might not be sustainable.

- Other techniques might not simply distinguish between friends and the rest of the network, but consider the degree of separation. This approach might potentially build a "chain of trust" that connects distant peers. Due to the small world topology of social networks, information about users' behaviour might travel fast through the network. In our future research, we will either consider real social networks, or generate graphs whose degree follows a power-law and that become more dense over time, as suggested by the forest fire model [47].

- Friend peers that are interested in the same content might form coalitions to download different sets of chunks, and dedicate part of their bandwidth for internal exchange of what they have downloaded.

As far as our credit-based model is concerned, we will formulate it within a game theoretic framework by considering the peer strategies and for example by proving that the system is stable in terms of Nash Equilibrium. Furthermore, an idea to improve the current state-of-the-art might come from Bayesian game theory [8]. In fact, each peer has a maximum upload capacity it wants to share (zero for a free-rider), which determines its maximum number of quality layers it can upload in real-time and therefore its type θ. Each peer knows its own type – which we can also assume to be assigned by Nature – and tries to estimate those of its neighbours from their actions. In order to achieve

a Perfect Bayesian Equilibrium (PBE) at each time every player must follow a strategy that maximises its expected pay-off given the expected strategies played by the other users. These strategies depend on beliefs, based on a peer's (incomplete) information set (e.g. a list of previous interactions). That is, at each round of the game a peer can follow a strategy that depends on the estimated type of its neighbour. Since we are dealing with transmission of scalable video streams, a peer might for example refuse to share enhancement layers of a sequence if it believes that its neighbour is not sharing enough bandwidth. This approach still presents some open issues, such as how to efficiently estimate a peer's type and what the optimal resource allocation policy is if a peer receives more requests than it can afford to accept.

Another open issue is the impact of sudden variations in the network (flash crowd, churn, etc.). Auction-based mechanisms [124] or other models which see P2P network as a market [125] could help solve this problem and this could become an interesting area of research in the near future. Finally, another area of research could be network coding [126,127] applied to P2P, which is an area we have already started exploring and our contribution can be found in [18].

Bibliography

[1] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. of 3rd Int. Conf. Peer-to-Peer Computing*, April 2003.

[2] S. Jun and M. Ahamad, "Incentives in BitTorrent induce free riding," in *Proc. of ACM SIGCOMM Workshop on Economics of peer-to-peer systems (P2PECON)*, pp. 116–121, 2005.

[3] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in P2P file sharing systems," in *Proc. of IEEE Conf. on Computer Communications*, December 2005.

[4] N. Ahmed, T. Natarajan, and K. Rao, "Discrete Cosine Transform," *Computers, IEEE Transactions on*, vol. C-23, no. 1, pp. 90–93, 1974.

[5] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, pp. 1103–1120, Sept. 2007.

[6] C. K. Chui, *An introduction to wavelets*. Academic Press Professional, Inc., 1992.

[7] N. Ramzan, T. Zgaljic, and E. Izquierdo, "An efficient optimisation scheme for scalable surveillance centric video communications," *Signal Processing: Image Communication*, vol. 24, pp. 510–523, July 2009.

[8] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. MIT Press, 1994.

[9] J. J. D. Mol, J. A. Pouwelse, M. Meulpolder, D. H. J. Epema, and H. J. Sips, "Give-to-get: free-riding resilient video-on-demand in P2P systems," in *Proc. of SPIE, Multimedia Computing and Networking Conference (MMCN)*, 2008.

[10] J. R. Douceur, "The Sybil Attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pp. 251–260, 2002.

[11] R. Landa, D. Griffin, R. G. Clegg, E. Mykoniati, and M. Rio, "A Sybilproof Indirect Reciprocity Mechanism for Peer-to-Peer Networks," in *Proceedings of IEEE INFOCOM 2009*, pp. 343–351, IEEE, 2009.

[12] N. Ramzan, E. Quacchio, T. Zgaljic, S. Asioli, L. Celetto, E. Izquierdo, and F. Rovati, "Peer-to-peer Streaming of Scalable Video in Future Internet Applications," *IEEE Communications Magazine, Special Issue on Future Media Internet*, vol. 49, pp. 128–135, March 2011.

[13] S. Asioli, N. Ramzan, and E. Izquierdo, "A Game Theoretic Approach to Minimum-delay Scalable Video Transmission over P2P," *Signal Processing: Image Communication*, vol. 27, no. 5, pp. 513–521, 2012.

[14] S. Asioli, N. Ramzan, and E. Izquierdo, "Efficient Scalable Video Streaming Over P2P Network," in *Proceedings of the 1st International ICST Conference on User Centric Media (UCMedia)*, December 2009.

[15] S. Asioli, N. Ramzan, and E. Izquierdo, "A Novel Technique for Efficient Peer-to-Peer Scalable Video Transmission," in *Proceedings of the 2010 European Signal Processing Conference (EUSIPCO)*, August 2010.

[16] S. Asioli, N. Ramzan, and E. Izquierdo, "A Game Theoretic Framework for Optimal Resource Allocation in P2P Scalable Video Streaming," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2293–2296, IEEE, 2012.

[17] S. Asioli, N. Ramzan, and E. Izquierdo, "Exploiting Social Relationships for Free-riders Detection in Minimum-delay P2P Scalable Video Streaming," in *Proceedings of the 2012 IEEE International Conference on Image Processing (ICIP)*, pp. 2257–2260, IEEE, 2012.

[18] R. Mekuria, M. Sanna, S. Asioli, E. Izquierdo, D. Bulterman, and P. Cesar, "A 3D Tele-Immersion System Based on Live Captured Mesh Geometry," in *Proceedings of the ACM Multimedia Systems Conference (ACM MMSys)*, 2013.

[19] N. Crespi, B. Molina, C. Palau, *et al.*, "QoE Aware Service Delivery in Distributed Environment," in *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pp. 837–842, IEEE, 2011.

[20] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "A quest for an Internet video quality-of-experience metric," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, HotNets-XI, pp. 97–102, ACM, 2012.

[21] R. Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications," in *Proceedings of the First International Conference on Peer-to-Peer Computing*, P2P '01, pp. 101–108, IEEE Computer Society, 2001.

[22] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI at home: an experiment in public-resource computing," *Commun. ACM*, vol. 45, pp. 56–61, Nov. 2002.

[23] S. A. Baset and H. Schulzrinne, "An analysis of the Skype peer-to-peer internet telephony protocol," in *IEEE infocom*, vol. 6, pp. 23–29, 2006.

[24] P. K. Gummadi, S. Saroiu, and S. D. Gribble, "A measurement study of Napster and Gnutella as examples of peer-to-peer file sharing systems," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 82–82, Jan. 2002.

[25] I. Filali, F. Bongiovanni, F. Huet, and F. Baude, "A survey of structured P2P systems for RDF data storage and retrieval," in *Transactions on large-scale data- and knowledge-centered systems III*, pp. 20–55, Springer, 2011.

[26] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "Deconstructing the KaZaA network," in *Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on*, pp. 112–120, IEEE, 2003.

[27] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in BitTorrent?," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, NSDI'07, (Berkeley, CA, USA), USENIX Association, 2007.

[28] E. Peserico, "P2P Economies," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2006.

[29] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing," in *Workshop on Economics*

of Peer-to-Peer Systems, 2003.

[30] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pp. 144–152, ACM, 2003.

[31] T. Zgalkic, M. Mrak, S. Wan, and N. Ramzan, "D3.11 Evaluation of aceSVC and solution for video playing," *aceMedia Technical Document*, 2007.

[32] T. Zgaljic, N. Sprljan, and E. Izquierdo, "Bit-stream allocation methods for scalable video coding supporting wireless communications," *Signal Processing: Image Communications*, vol. 22, no. 3, pp. 298–316, 2007.

[33] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 559–571, 1994.

[34] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445–3462, December 1993.

[35] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, pp. 560–576, July 2003.

[36] "ITU-T H.262 , Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video, International Telecommunication Union Recommendation, 1994."

[37] G. Ct, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bitrates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 849–866, 1998.

[38] C. Poynton, "Chroma subsampling notation," *Digital Video and HDTV: Algorithms and Interfaces*, vol. 19, p. 2004, 2002.

[39] N. Ljepava, R. R. Orr, S. Locke, and C. Ross, "Personality and social characteristics of facebook non-users and frequent users," *Computers in Human Behavior*, vol. 29, no. 4, pp. 1602–1607, 2013.

[40] N. B. Ellison *et al.*, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[41] A. Papanikolaou, V. Vlachos, P. Chatzimisios, and C. Ilioudis, "Privacy Issues in Social Networks," *Social Network Engineering for Secure Web Data and Services*, p. 162, 2013.

[42] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, June 1998.

[43] S. Milgram, "The Small World Problem," *Psychology Today*, vol. 1, no. 1, pp. 61–67, 1967.

[44] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna, "Four degrees of separation," in *Proceedings of the 3rd Annual ACM Web Science Conference*, WebSci '12, pp. 33–42, ACM, 2012.

[45] V. Batagelj and A. Mrvar, "Pajek – Program for Analysis and Visualization of Large Networks, Reference Manual." 2011.

[46] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[47] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 2, 2007.

[48] R. S. Cruz, M. S. Nunes, C. Z. Patrikakis, and N. C. Papaoulakis, "SARACEN: A platform for adaptive, Socially Aware, collaboRative, scAlable Coding mEdia distributioN," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pp. 1356–1360, IEEE, 2010.

[49] M. Tekalp, N. Ramzan, C. G. Gurler, S. Asioli, T. Bagci, and D. Esteban, "D3.1 Evaluation of scalable and multiple description coding in P2P applications," *Saracen Technical Document*, 2012.

[50] M. Tekalp, N. Ramzan, C. G. Gurler, S. Asioli, T. Bagci, C. Patrikakis, N. Papaoulakis, and N. Protonotarios, "D3.2 Measuring and Optimization of Quality of Experience in P2P video streaming," *Saracen Technical Document*, 2013.

[51] M. Tekalp, N. Ramzan, T. Bagci, S. Asioli, C. G. Gurler, E. Garrido, and D. Esteban, "D3.4 Evaluation of the Resilience Mechanisms For Integration in

the Proposed Architecture," *Saracen Technical Document*, 2013.

[52] R. Cruz, M. Nunes, N. Ramzan, S. Asioli, and D. Esteban, "D4.2 Evaluation of Adaptive and QoS aware scalable coded streaming mechanism," *Saracen Technical Document*, 2012.

[53] S. Winkler and P. Mohandas, "The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics," *Broadcasting, IEEE Transactions on*, vol. 54, no. 3, pp. 660–668, 2008.

[54] B. Girod, "What's wrong with mean-squared error?," in *Digital images and human vision*, pp. 207–220, MIT press, 1993.

[55] J. Guo, M. Van Dyke-Lewis, and H. Myler, "Gabor difference analysis of digital video quality," *Broadcasting, IEEE Transactions on*, vol. 50, no. 3, pp. 302–311, 2004.

[56] S. S. Hemami and A. R. Reibman, "No-reference image and video quality estimation: Applications and human-motivated design," *Image Commun.*, vol. 25, pp. 469–481, Aug. 2010.

[57] J. Monteiro and M. S. Nunes, "A subjective quality estimation tool for the evaluation of video communication systems," in *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on*, pp. 75–80, 2007.

[58] "ITU-R BT.500-12, Methodology for the subjective assessment of the quality of television pictures, International Telecommunication Union Recommendation, Rev. 12, 2009.."

[59] "ITU-T P.910, Subjective video quality assessment methods for multimedia applications, International Telecommunication Union Recommendation, 1999.."

[60] A. Puri and A. Eleftheriadis, "MPEG-4: an object-based multimedia coding standard supporting mobile applications," *Mob. Netw. Appl.*, vol. 3, pp. 5–32, June 1998.

[61] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *International Journal of Human-Computer Studies*, vol. 64, pp. 637–647, Aug. 2006.

[62] G. Zhai, J. Cai, W. Lin, X. Yang, W. Zhang, and M. Etoh, "Cross-dimensional perceptual quality assessment for low bit-rate videos," *IEEE Transactions on Multimedia*, vol. 10, no. 7, pp. 1316–1324, 2008.

[63] A. Eichhorn and P. Ni, "Pick your layers wisely - a quality assessment of H.264 scalable video coding for mobile devices," in *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, pp. 5446–5451, 2009.

[64] W. Song, D. Tjondronegoro, and S. Azad, "User-Centered video quality assessment for scalable video coding of H.264/AVC standard," in *Proceedings of the 16th international conference on Advances in Multimedia Modeling*, MMM'10, pp. 55–65, 2010.

[65] J.-S. Lee, F. De Simone, N. Ramzan, Z. Zhao, E. Kurutepe, T. Sikora, J. Ostermann, E. Izquierdo, and T. Ebrahimi, "Subjective evaluation of scalable video coding for content distribution," in *Proceedings of the International ACM conference on Multimedia*, MM '10, pp. 65–72, 2010.

[66] C. Fenimore, J. Libert, and S. Wolf, "Perceptual effects of noise in digital video compression," in *140th SMPTE Technical Conference*, pp. 28–31, 1998.

[67] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "QDASH: a QoE-aware DASH system," in *Proceedings of the 3rd Multimedia Systems Conference*, pp. 11–22, ACM, 2012.

[68] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips, "Tribler: A social-based peer-to-peer system," *Concurrency and Computation: Practice and Experience Journal*, vol. 20, pp. 127–138, February 2008.

[69] Y. Huang, T. Z. Fu, D.-M. Chiu, J. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P-VoD system," in *ACM SIGCOMM Computer Communication Review*, pp. 375–388, ACM, 2008.

[70] N. Vratonjić, P. Gupta, N. Knežević, D. Kostić, and A. Rowstron, "Enabling DVD-like features in P2P Video-on-Demand Systems," in *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, pp. 329–334, ACM, 2007.

[71] S. Annapureddy, C. Gkantsidis, P. Rodriguez, and L. Massoulie, "Providing Video-on-Demand using Peer-to-Peer networks," in *Proc. Internet Protocol TeleVision (IPTV) workshop*, 2006.

[72] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," in *ACM SIGOPS Operating Systems Review*, pp. 298–313, ACM, 2003.

[73] I. Lee and L. Guan, "Reliable video communication with multi-path streaming using MDC," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pp. 4–pp, IEEE, 2005.

[74] P. Shah and J. F. Paris, "Peer-to-peer multimedia streaming using BitTorrent," in *Proc. of IEEE International Performance, Computing, and Communications Conference, 2007. IPCCC 2007.*, 2007.

[75] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," in *Proc. of 25th IEEE International Conference on Computer Communications IEEE (INFOCOMM)*, April 2006.

[76] T. Tucker, D. Pate, and C. Rattanapoka, "ABC [Yet Another Bittorrent Client]."

[77] J. Hoffman, "BitTornado," 2011.

[78] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Using layered video to provide incentives in P2P live streaming," in *Proc. of 2007 Workshop on Peer-to-peer streaming and IP-TV (P2P-TV)*, pp. 311–316, 2007.

[79] X. Hei, Y. Liu, and K. W. Ross, "IPTV over P2P streaming networks: the mesh-pull approach," *IEEE Communications Magazine*, vol. 46, pp. 86–92, March 2008.

[80] S. Xie, B. Li, G. Keung, and X. Zhang, "Coolstreaming: Design, Theory, and Practice," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1661–1671, 2007.

[81] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in BitTorrent systems," in *ACM SIGMETRICS Performance Evaluation Review*, pp. 301–312, ACM, 2007.

[82] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica, 1995.

[83] R. A. Adams, *Calculus : A Complete Course*. Addison-Wesley, 5th ed., 2003.

[84] C. Buragohain, D. Agrawal, and S. Suri, "A game theoretic framework for incentives in P2P systems," in *Proc. of 3rd International Conference on Peer-to-Peer Computing*, 2003.

[85] W. S. Lin, H. V. Zhao, and K. J. R. Liu, "Incentive cooperation strategies for peer-to-peer live multimedia streaming social networks," *IEEE Trans. on Multimedia*, vol. 11, pp. 396–412, April 2009.

[86] K. Aberer and Z. Despotovic, "On reputation in game theory - Application to online settings," tech. rep., Technical Report, 2004.

[87] G. R. Grimmett and D. R. Stirzaker, *Probability and random processes*. Oxford University Press, USA, 2001.

[88] H. Bergström, "On the central limit theorem," *Scandinavian Actuarial Journal*, vol. 1944, no. 3-4, pp. 139–153, 1944.

[89] J. Park and M. van der Schaar, "A game theoretic analysis of incentives in content production and sharing over peer-to-peer networks," *Computing Research Repository*, October 2009.

[90] H. Park and M. van der Schaar, "A framework for foresighted resource reciprocation in P2P networks," *IEEE Trans. on Multimedia*, vol. 11, January 2009.

[91] H. Park and M. van der Schaar, "Evolution of resource reciprocation strategies in P2P networks," *IEEE Trans. on Signal Processing*, vol. 58, March 2010.

[92] L. Chisci, F. Papi, T. Pecorella, and R. Fantacci, "An evolutionary game approach to P2P video streaming," in *Proc. of IEEE GLOBECOM*, pp. 1–5, 2009.

[93] Y. Matsuda, M. Sasabe, and T. Takine, "Evolutionary game theory-based evaluation of P2P file-sharing systems in heterogeneous environments," *International Journal of Digital Multimedia Broadcasting*, 2010.

[94] Y. Chen, B. Wang, W. Lin, Y. Wu, and K. Liu, "Cooperative peer-to-peer streaming: an evolutionary game-theoretic approach," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 20, no. 10, pp. 1346–1357, 2010.

[95] C. Aperjis and R. Johari, "A peer-to-peer system as an exchange economy," in *Proc. of 2006 Workshop on Game theory for communications and networks (GameNets)*, 2006.

[96] D. P. Bertsekas, *Dynamic Programming and Stochastic Control*. New York: Academic, 1976.

[97] J. W. Weibull, *Evolutionary Game Theory*. MIT Press, 1997.

[98] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proc. of Conf. on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM)*, pp. 251–262, 1999.

[99] C. E. Tsourakakis, P. Drineas, E. Michelakis, I. Koutis, and C. Faloutsos, "Spectral counting of triangles in power-law networks via element-wise sparsification," in *Proc. of Int. Conf. on Advances in Social Network Analysis and Mining (ASONAM)*, pp. 66–71, IEEE Computer Society, 2009.

[100] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, pp. 814–818, June 2005.

[101] S. Fortunato, A. Flammini, and F. Menczer, "Scale-free network growth by ranking," *Physical Review Letters*, vol. 96, no. 21, 2006.

[102] L. Zhang, J. K. Muppala, and W. Tu, "Exploiting proximity in cooperative download of large files in peer-to-peer networks," in *Proc. of 2nd International Conference on Internet and Web Applications and Services (ICIW)*, 2007.

[103] B. Liu, Y. Cui, Y. Lu, and Y. Xue, "Locality-awareness in BitTorrent-like P2P applications," *IEEE Trans. on Multimedia*, vol. 11, pp. 361–371, April 2009.

[104] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, "Leveraging social networks for increased BitTorrent robustness," in *Proc. of 7th IEEE Conference on Consumer Communications and Networking*, pp. 159–163, 2010.

[105] F. Huang, B. Ravindran, and A. Vullikanti, "An approximation algorithm for minimum-delay peer-to-peer streaming," in *Proc. of Int. Conf. on Peer-to-Peer Computing*, pp. 71–80, September 2009.

[106] F. Picconi and L. Massoulie, "Is there a future for mesh-based live video streaming?," in *Proc. of 8th Int. Conf. on Peer-to-Peer Computing*, pp. 289–298, 2008.

[107] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized Decentralized Broadcasting Algorithms," in *Proc. of 26th IEEE Int. Conf. on Computer Communications, INFOCOM 2007*, pp. 1073–1081, 2007.

[108] P. Baccichet, T. Schierl, T. Wiegand, and B. Girod, "Low-delay peer-to-peer streaming using scalable video coding," in *Proc. of Packet Video 2007*, pp. 173–181, 2007.

[109] C. Liang, Y. Guo, and Y. Liu, "Investigating the scheduling sensitivity of P2P video streaming: an experimental study," *IEEE Trans. on Multimedia*, vol. 11, pp. 348–360, April 2009.

[110] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. on Multimedia*, vol. 9, pp. 1672–1687, November 2007.

[111] "ns-2, Network Simulator 2," 1989.

[112] K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann, "Efficient simulation of large-scale P2P networks: packet-level vs. flow-level simulations," in *Proc. of 2nd Workshop on Use of P2P, GRID and Agents for the Development of Content Networks*, UPGRADE '07, pp. 9–16, ACM, 2007.

[113] B. Stroustrup, *The C++ Programming Language*. Addison-Wesley, 2000.

[114] J. K. Ousterhout and K. Jones, *Tcl and the Tk toolkit*, vol. 227. Addison-Wesley, 1994.

[115] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "PeerSoN: P2P social networking: early experiences and insights," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pp. 46–52, ACM, 2009.

[116] Y.-F. Wang, Y. Hori, and K. Sakurai, "Characterizing economic and social properties of trust and reputation systems in P2P environment," *Journal of Computer Science and Technology*, vol. 23, no. 1, pp. 129–140, 2008.

[117] Y. Zhang and M. van der Schaar, "Peer-to-Peer multimedia sharing based on social norms," *Signal Processing: Image Communications*, vol. 27, pp. 383–400, May 2012.

[118] A. M. Manago, T. Taylor, and P. M. Greenfield, "Me and my 400 friends: The anatomy of college students' Facebook networks, their communication patterns, and well-being.," *Developmental psychology*, vol. 48, no. 2, p. 369, 2012.

[119] H. Yu, M. Kaminsky, P. B. Gibbons, and A. D. Flaxman, "SybilGuard: Defending against sybil attacks via social networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 576–589, 2008.

[120] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, 2012.

[121] "ITU-T H.265, High Efficiency Video Coding, International Telecommunication Union Recommendation, 2013."

[122] D. Hong, W. Jang, J. Boyce, and A. Abbas, "Scalability support in hevc," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pp. 890–893, 2012.

[123] Z. Shi, X. Sun, and F. Wu, "Spatially scalable video coding for hevc," in *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pp. 1091–1096, 2012.

[124] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee, "Bittorrent is an Auction: Analyzing and Improving BitTorrent's Incentives," in *Proc. of the ACM SIGCOMM 2008 Conf. on Data communication*, SIGCOMM '08, pp. 243–254, ACM, 2008.

[125] A. H. Payberah, F. Rahimian, S. Haridi, and J. Dowling, "Sepidar: Incentivized Market-Based P2P Live-Streaming on the Gradient Overlay Network," *IEEE International Symposium on Multimedia*, pp. 1–8, 2010.

[126] M. Sanna and E. Izquierdo, "A Method for Detection/Deletion via Network Coding for Unequal Error Protection of Scalable Video over Error-Prone Networks," in *Mobile Multimedia Communications*, pp. 105–120, Springer, 2012.

[127] H. Zeng, J. Huang, S. Tao, and W. Cheng, "A simulation study on network coding parameters in P2P content distribution system," in *Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on*, pp. 197–201, IEEE, 2008.